In [133…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
```

In [134…
```
data=pd.read_csv('Teleco_Customer_Churn.csv')
```

In [135…
```
data.head()
```

Out[135…

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No

5 rows × 21 columns

In [136…
```
data.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
```

In [137…
```
# to identify no.of rows and columns
data.shape
```

Out[137…
```
(7043, 21)
```

In [138…
```
# check for any null vales in data
data.isnull().any()
```

Out[138…
```
customerID          False
gender              False
SeniorCitizen       False
Partner             False
Dependents          False
tenure              False
PhoneService        False
MultipleLines       False
InternetService     False
OnlineSecurity      False
OnlineBackup        False
DeviceProtection    False
TechSupport         False
StreamingTV         False
StreamingMovies     False
Contract            False
PaperlessBilling    False
PaymentMethod       False
MonthlyCharges      False
TotalCharges        False
Churn               False
dtype: bool
```

In [139…
```
data.isnull().sum()
```

Out[139…
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

In [140…
```
# check for duplicates
duplicates=data.duplicated()
print(duplicates.sum())
```

```
0
```

In [141…
```
data.isnull().values.any()
```

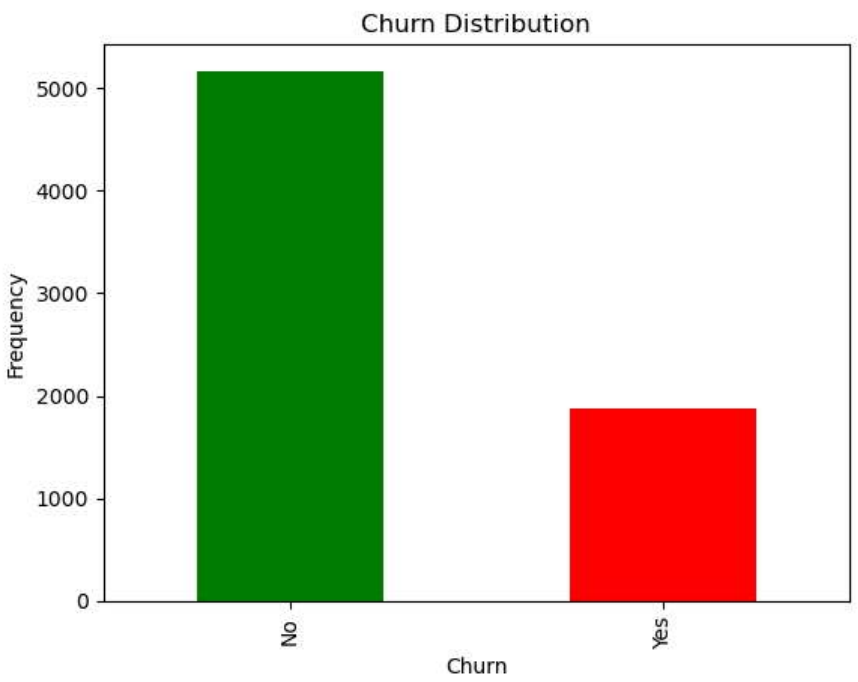
Out[141…
```
False
```

In [142…
```
data['TotalCharges']=pd.to_numeric(data['TotalCharges'],errors='coerce')
total=data['TotalCharges'].sum()
print(total)
```

```
16056168.7
```

In [143…
```
# univariant analysis
# Distribution of churn (Yes/No).
churn_distribution=data['Churn'].value_counts()
print(churn_distribution)
data['Churn'].value_counts().plot(kind='bar', color=['green', 'red'])
plt.title('Churn Distribution')
plt.xlabel('Churn')
plt.ylabel('Frequency')
plt.show()
```

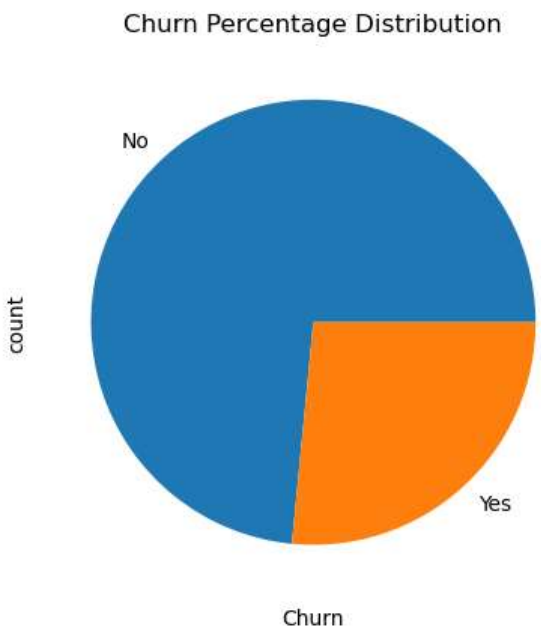
```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
```



In [144…

```
# pie chart in terms of percentage
churn_count=data['Churn'].value_counts()
churn_yespercent=(churn_count['Yes']/churn_count.sum())*100
churn_nopercent=(churn_count['No']/churn_count.sum())*100
print(churn_count)
print(f"Churn Yes Percentage: {churn_yespercent:.2f}%")
print(f"Churn No Percentage: {churn_nopercent:.2f}%")
data['Churn'].value_counts().plot(kind='pie',color=['green','red'],labels=churn_count.index)
plt.title('Churn Percentage Distribution')
plt.xlabel('Churn')
plt.show()
```

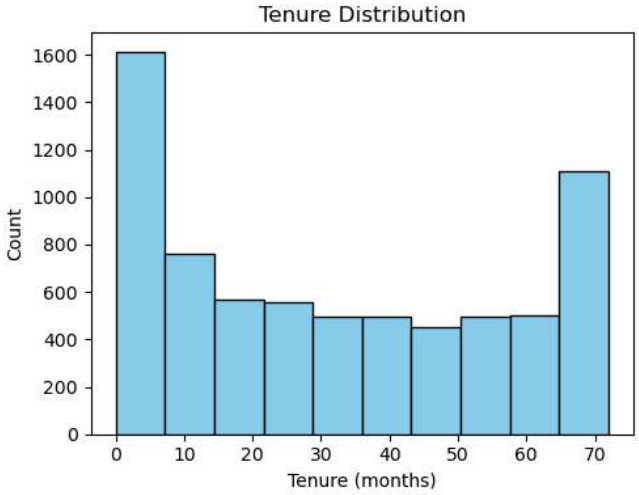
```
Churn
No     5174
Yes    1869
Name: count, dtype: int64
Churn Yes Percentage: 26.54%
Churn No Percentage: 73.46%
```

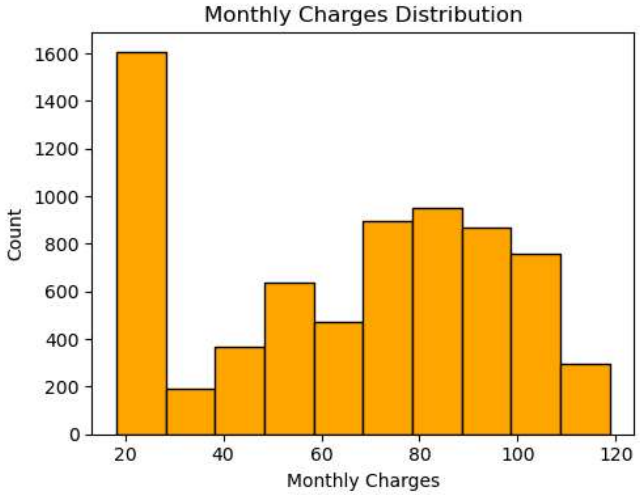


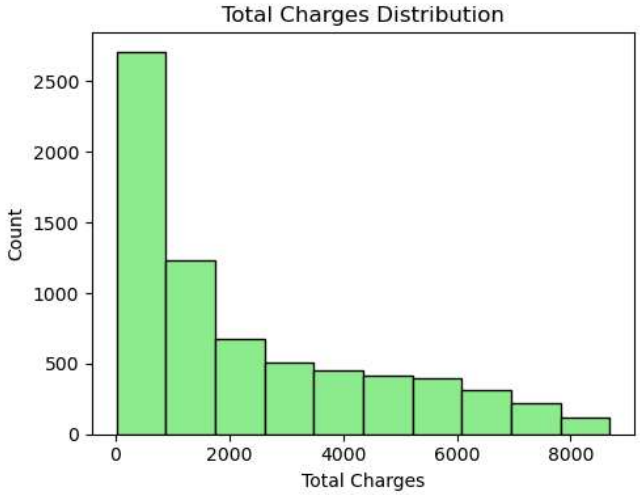
In [145…

```ipynb
#Histograms of numerical features (tenure, MonthlyCharges, TotalCharges).
data[['tenure','MonthlyCharges','TotalCharges']].info()
plt.figure(figsize=(15, 4))

# Tenure
plt.subplot(1, 3, 1)
plt.hist(data['tenure'].dropna(), color='skyblue', edgecolor='black')
plt.title('Tenure Distribution')
plt.xlabel('Tenure (months)')
plt.ylabel('Count')

# Monthly Charges
plt.subplot(1, 3, 2)
plt.hist(data['MonthlyCharges'].dropna(), color='orange', edgecolor='black')
plt.title('Monthly Charges Distribution')
plt.xlabel('Monthly Charges')
plt.ylabel('Count')

# Total Charges
plt.subplot(1, 3, 3)
plt.hist(data['TotalCharges'].dropna(), color='lightgreen', edgecolor='black')
plt.title('Total Charges Distribution')
plt.xlabel('Total Charges')
plt.ylabel('Count')

plt.tight_layout()
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   tenure          7043 non-null   int64  
 1   MonthlyCharges  7043 non-null   float64
 2   TotalCharges    7032 non-null   float64
dtypes: float64(2), int64(1)
memory usage: 165.2 KB
```

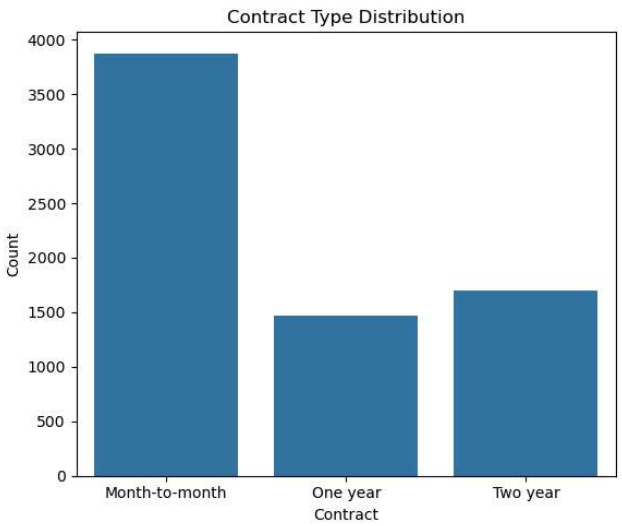


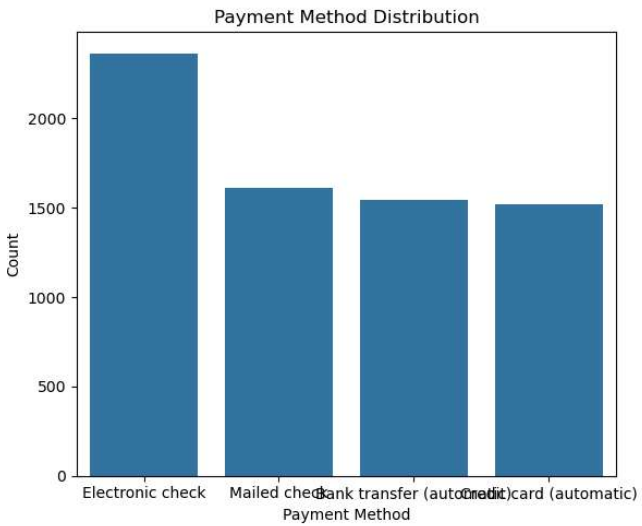


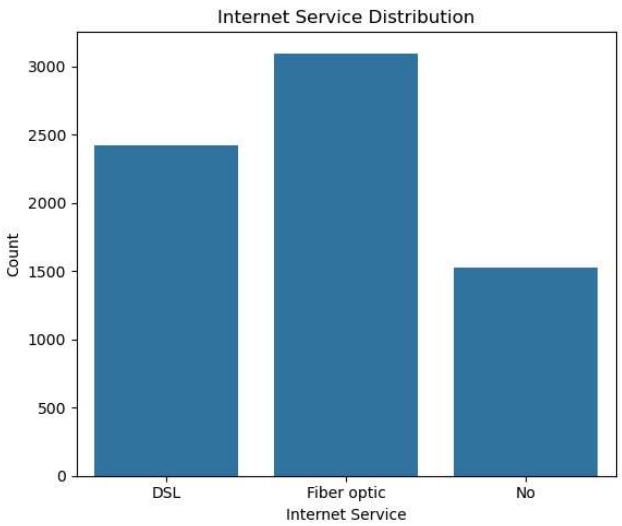


In [146…

```python
# Count plots for categorical features (Contract, PaymentMethod, InternetService).
plt.figure(figsize=(18, 5))

# Contract
plt.subplot(1, 3, 1)
sns.countplot(data=data, x='Contract')
plt.title('Contract Type Distribution')
plt.xlabel('Contract')
plt.ylabel('Count')

# Payment Method
plt.subplot(1, 3, 2)
sns.countplot(data=data, x='PaymentMethod')
plt.title('Payment Method Distribution')
plt.xlabel('Payment Method')
plt.ylabel('Count')

# Internet Service
plt.subplot(1, 3, 3)
sns.countplot(data=data, x='InternetService')
plt.title('Internet Service Distribution')
plt.xlabel('Internet Service')
plt.ylabel('Count')

plt.tight_layout()
plt.show()
```

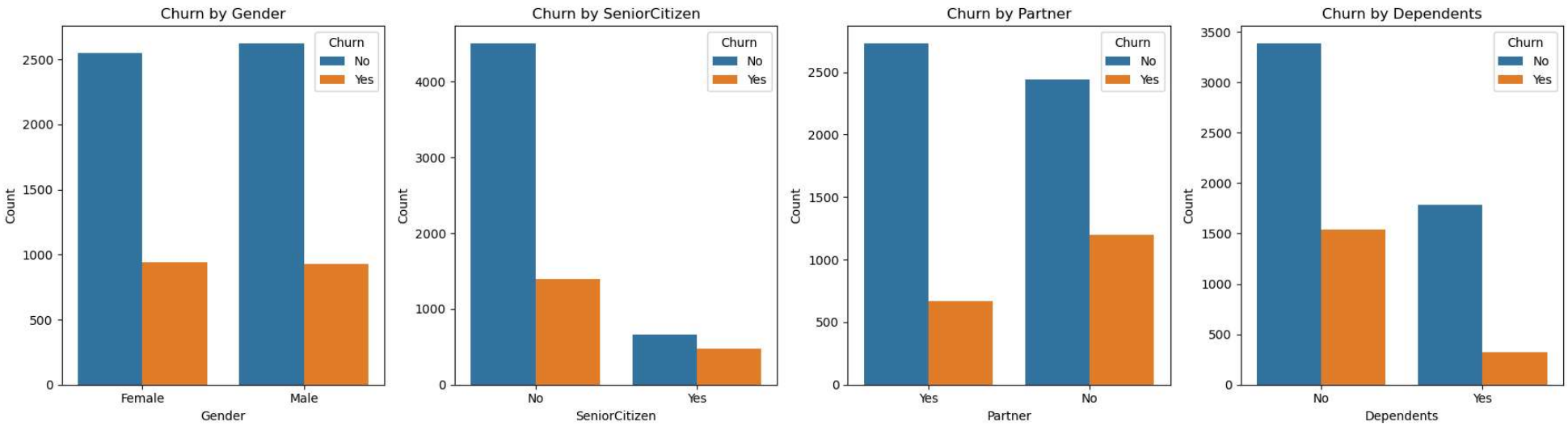






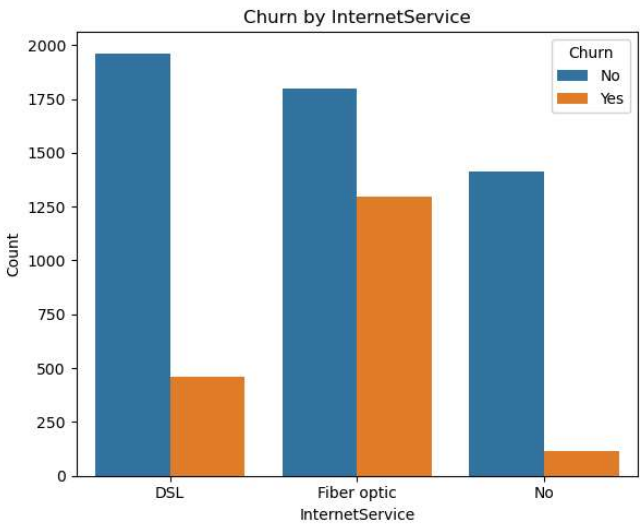
In [147…

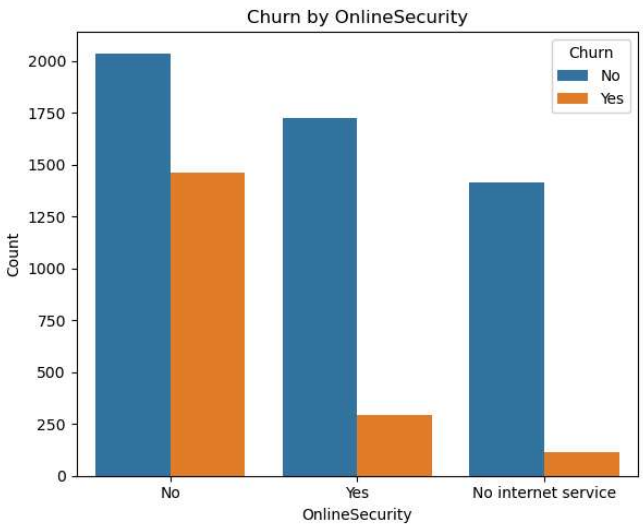
```
# Bivariate Analysis
# Churn by demographics (gender, SeniorCitizen, Partner, Dependents).
plt.figure(figsize=(18, 5))
plt.subplot(1,4,1)
sns.countplot(data=data, x='gender', hue='Churn')
plt.title('Churn by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')

plt.subplot(1,4,2)
data['SeniorCitizen']=data['SeniorCitizen'].map({0:'No',1:'Yes'})
sns.countplot(data=data, x='SeniorCitizen', hue='Churn')
plt.title('Churn by SeniorCitizen')
plt.xlabel('SeniorCitizen')
plt.ylabel('Count')

plt.subplot(1,4,3)
sns.countplot(data=data, x='Partner', hue='Churn')
plt.title('Churn by Partner')
plt.xlabel('Partner')
plt.ylabel('Count')

plt.subplot(1,4,4)
sns.countplot(data=data, x='Dependents', hue='Churn')
plt.title('Churn by Dependents')
plt.xlabel('Dependents')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

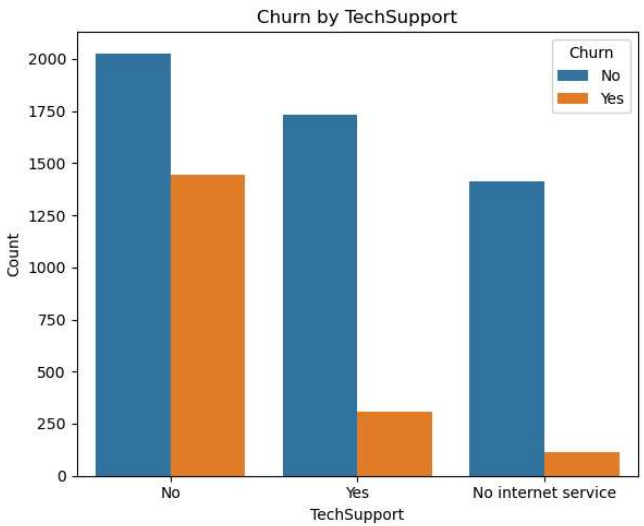


In [148…

```
#Churn by services (InternetService, OnlineSecurity, TechSupport).
plt.figure(figsize=(18, 5))
plt.subplot(1,3,1)
sns.countplot(data=data, x='InternetService', hue='Churn')
plt.title('Churn by InternetService')
plt.xlabel('InternetService')
plt.ylabel('Count')

plt.subplot(1,3,2)
sns.countplot(data=data, x='OnlineSecurity', hue='Churn')
plt.title('Churn by OnlineSecurity')
plt.xlabel('OnlineSecurity')
plt.ylabel('Count')

plt.subplot(1,3,3)
sns.countplot(data=data, x='TechSupport', hue='Churn')
plt.title('Churn by TechSupport')
plt.xlabel('TechSupport')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

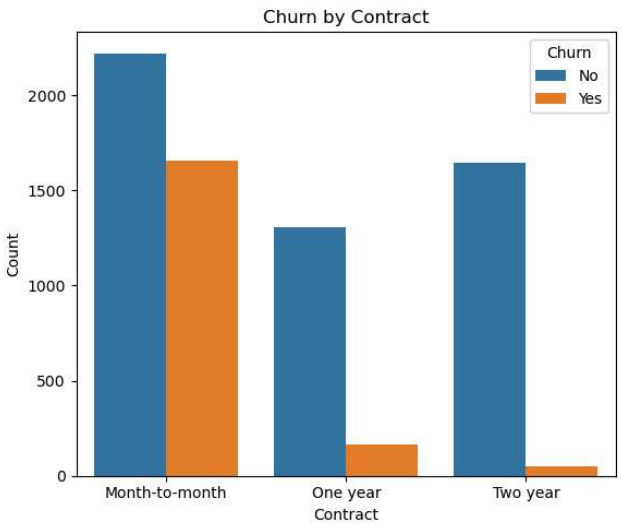


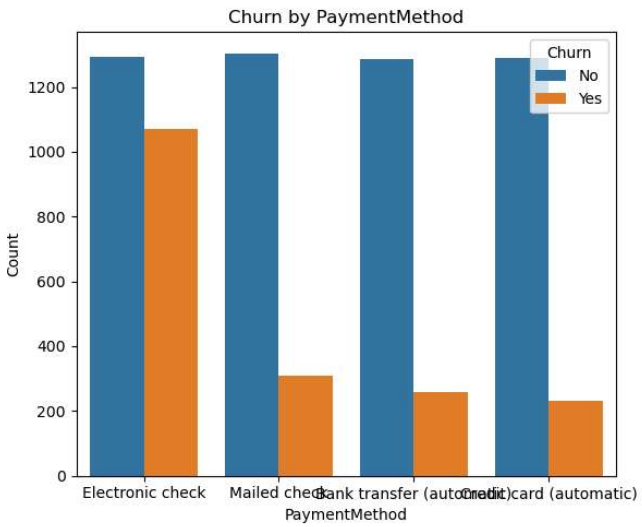


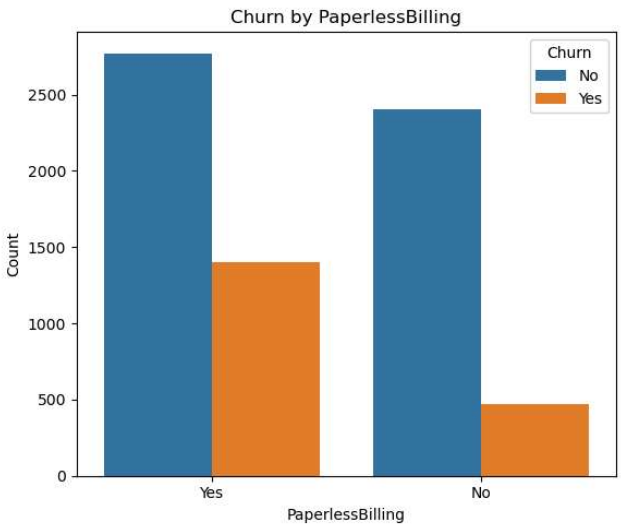


In [149…

```
#Churn by account details (Contract, PaymentMethod, PaperlessBilling).
plt.figure(figsize=(18, 5))
plt.subplot(1,3,1)
sns.countplot(data=data, x='Contract', hue='Churn')
plt.title('Churn by Contract')
plt.xlabel('Contract')
plt.ylabel('Count')

plt.subplot(1,3,2)
sns.countplot(data=data, x='PaymentMethod', hue='Churn')
plt.title('Churn by PaymentMethod')
plt.xlabel('PaymentMethod')
plt.ylabel('Count')

plt.subplot(1,3,3)
sns.countplot(data=data, x='PaperlessBilling', hue='Churn')
plt.title('Churn by PaperlessBilling')
plt.xlabel('PaperlessBilling')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

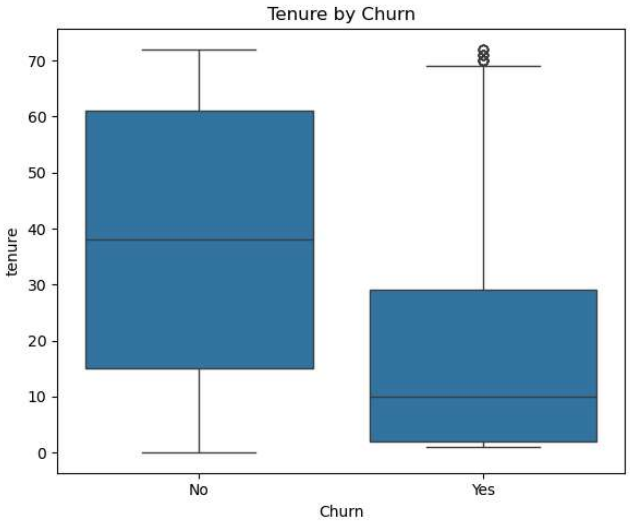


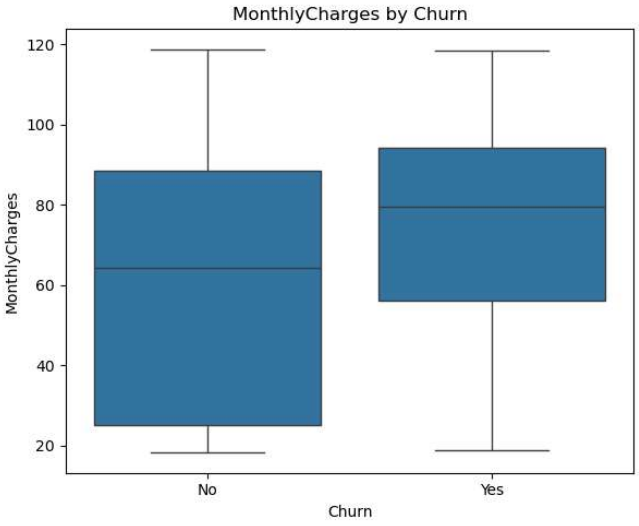


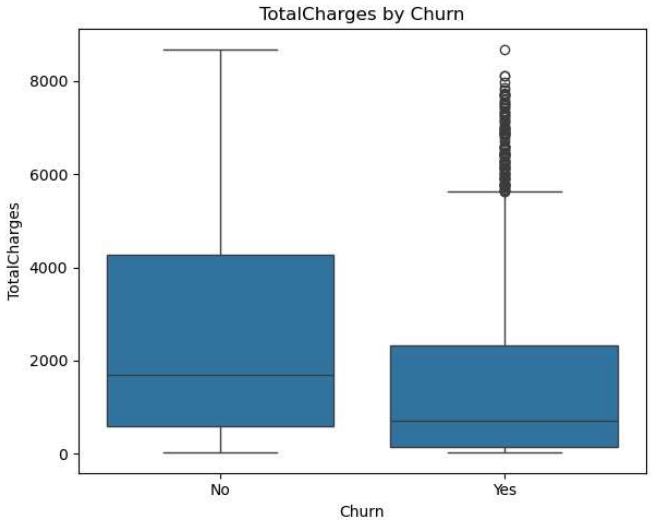


In [150…

```
#Box plots of tenure & charges by churn
plt.figure(figsize=(18, 5))
plt.subplot(1,3,1)
sns.boxplot(data=data, x='Churn', y='tenure')
plt.title('Tenure by Churn')

plt.subplot(1,3,2)
sns.boxplot(data=data, x='Churn', y='MonthlyCharges')
plt.title('MonthlyCharges by Churn')

plt.subplot(1,3,3)
sns.boxplot(data=data, x='Churn', y='TotalCharges')
plt.title('TotalCharges by Churn')
plt.tight_layout()
plt.show()
```



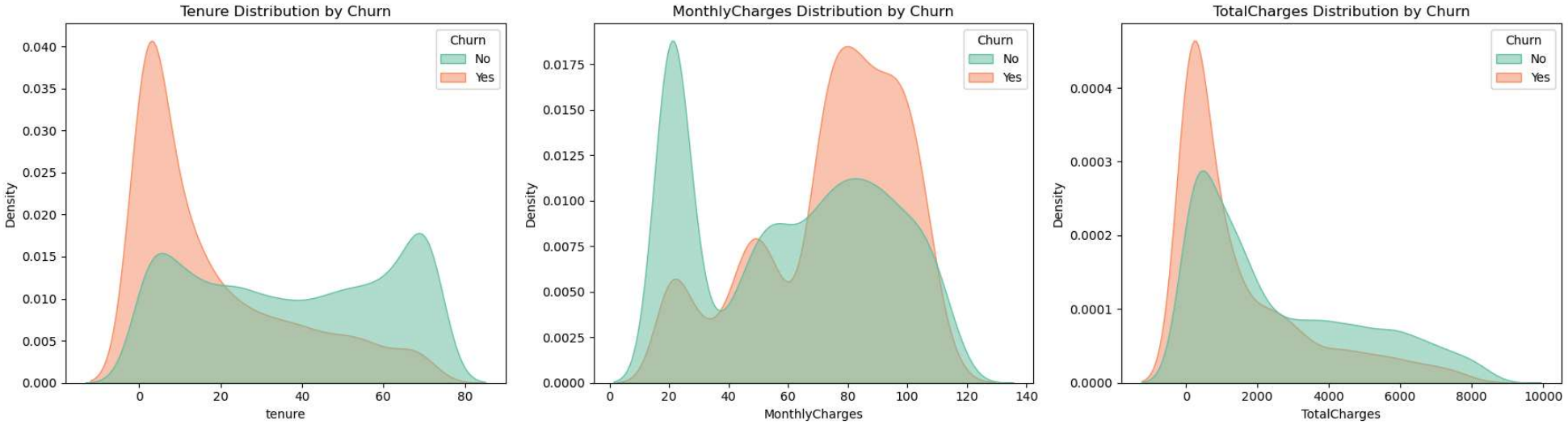




In [151…

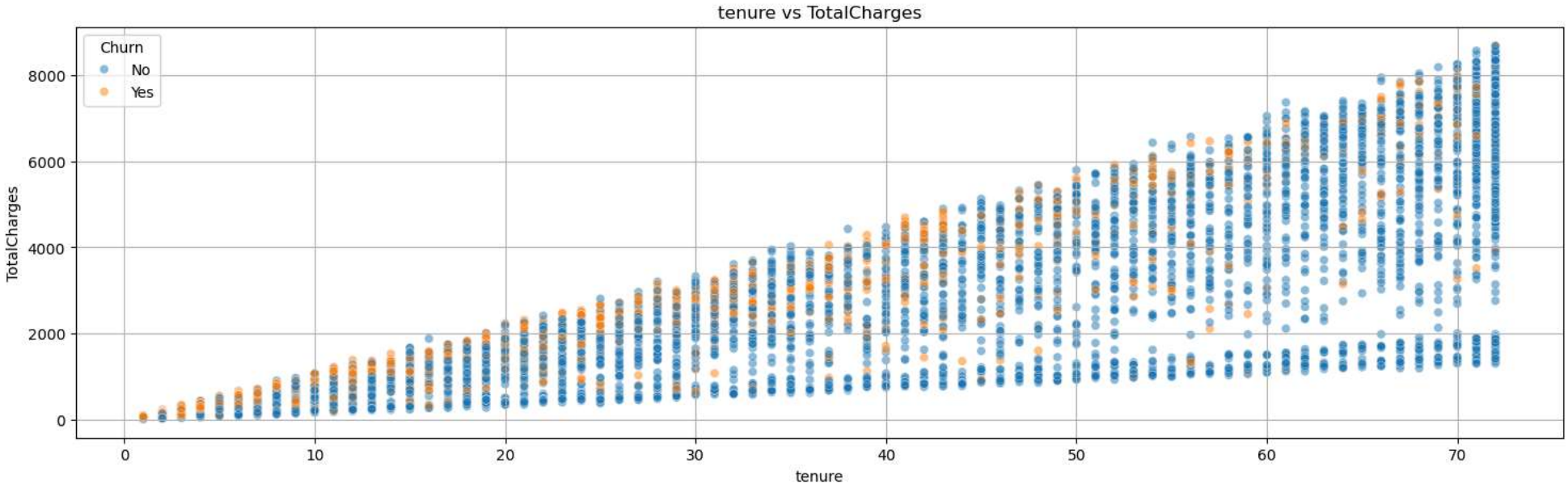
```
#KDE(Kernel Density Estimate) plots of tenure & charges by churn

plt.figure(figsize=(18,5))
plt.subplot(1,3,1)
```

```
sns.kdeplot(data=data, x='tenure', hue='Churn', fill=True, common_norm=False, palette='Set2', alpha=0.5)
plt.title('Tenure Distribution by Churn')

plt.subplot(1,3,2)
sns.kdeplot(data=data, x='MonthlyCharges', hue='Churn', fill=True, common_norm=False, palette='Set2', alpha=0.5)
plt.title('MonthlyCharges Distribution by Churn')

plt.subplot(1,3,3)
sns.kdeplot(data=data, x='TotalCharges', hue='Churn', fill=True, common_norm=False, palette='Set2', alpha=0.5)
plt.title('TotalCharges Distribution by Churn')
plt.tight_layout()
plt.show()
```



In [152…

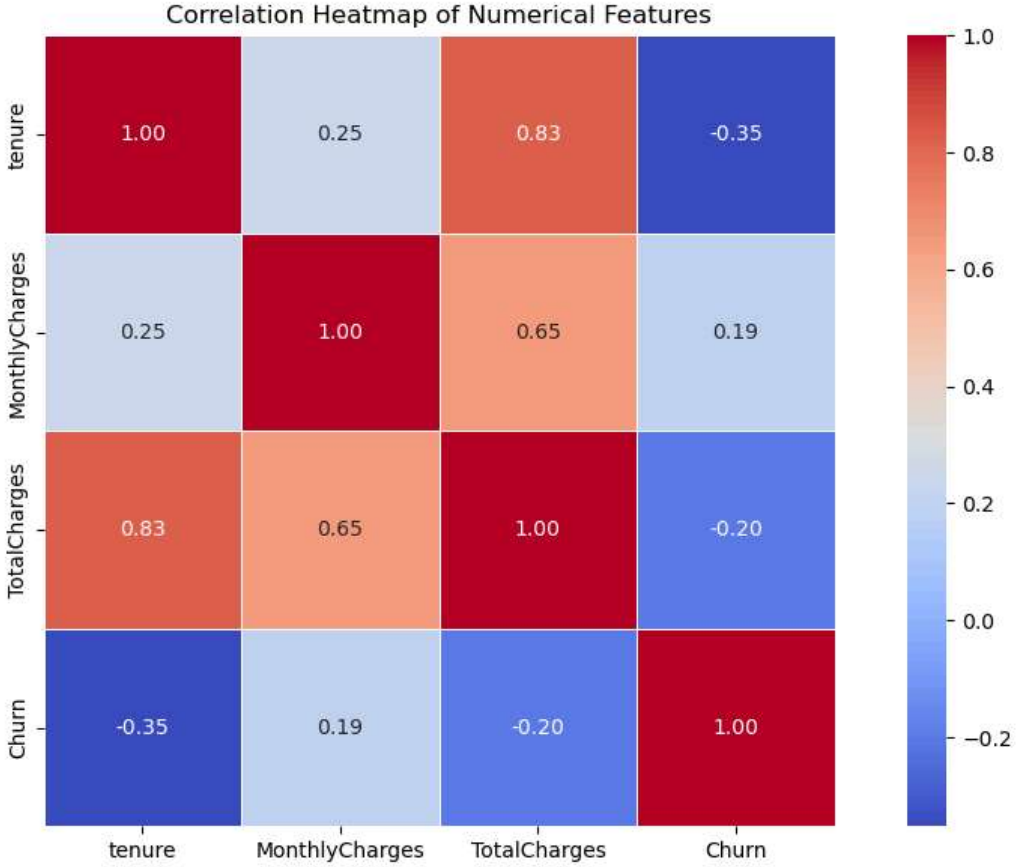
```
#Scatter plot: tenure vs. TotalCharges (color by churn).
plt.figure(figsize=(18,5))
sns.scatterplot(data=data,x='tenure',y='TotalCharges',hue='Churn',alpha=0.5)
plt.title('tenure vs TotalCharges')
plt.xlabel('tenure')
plt.ylabel('TotalCharges')
plt.grid(True)
plt.legend(title='Churn')
plt.show()

#Upward trend: Generally, longer tenure → higher total charges.
#Churn clusters: churned customers clustered in short-tenure, low-charge regions.
#Outliers: Look for any odd points (e.g., long-tenure with low charges).
```



In [153…

```
#Multivariate & Advanced Analysis
#Correlation heatmap of numerical features
data['Churn'] = data['Churn'].map({'Yes': 1, 'No': 0})
numeric_columns=data.select_dtypes(include=['float64','int64'])
print(numeric_columns)
corr_matrix = numeric_columns.corr()
print(corr_matrix)
plt.figure(figsize=(10, 6))
sns.heatmap(
    corr_matrix,
    annot=True,              # Show correlation values
    fmt='.2f',               # Format numbers to 2 decimal places
    cmap='coolwarm',         # Color map (blue = negative, red = positive)
    square=True,
    linewidths=0.5,
    #cbar_kws={'shrink': 0.7} # Shrink colorbar
)

plt.title('Correlation Heatmap of Numerical Features')
plt.tight_layout()
plt.show()

#Values near 1 → strong positive correlation
#Values near -1 → strong negative correlation
#Values near 0 → little or no correlation
#tenure and TotalCharges likely have a strong positive correlation
#MonthlyCharges and TotalCharges might also be somewhat correlated
#weak or no correlation with variables like SeniorCitizen
```

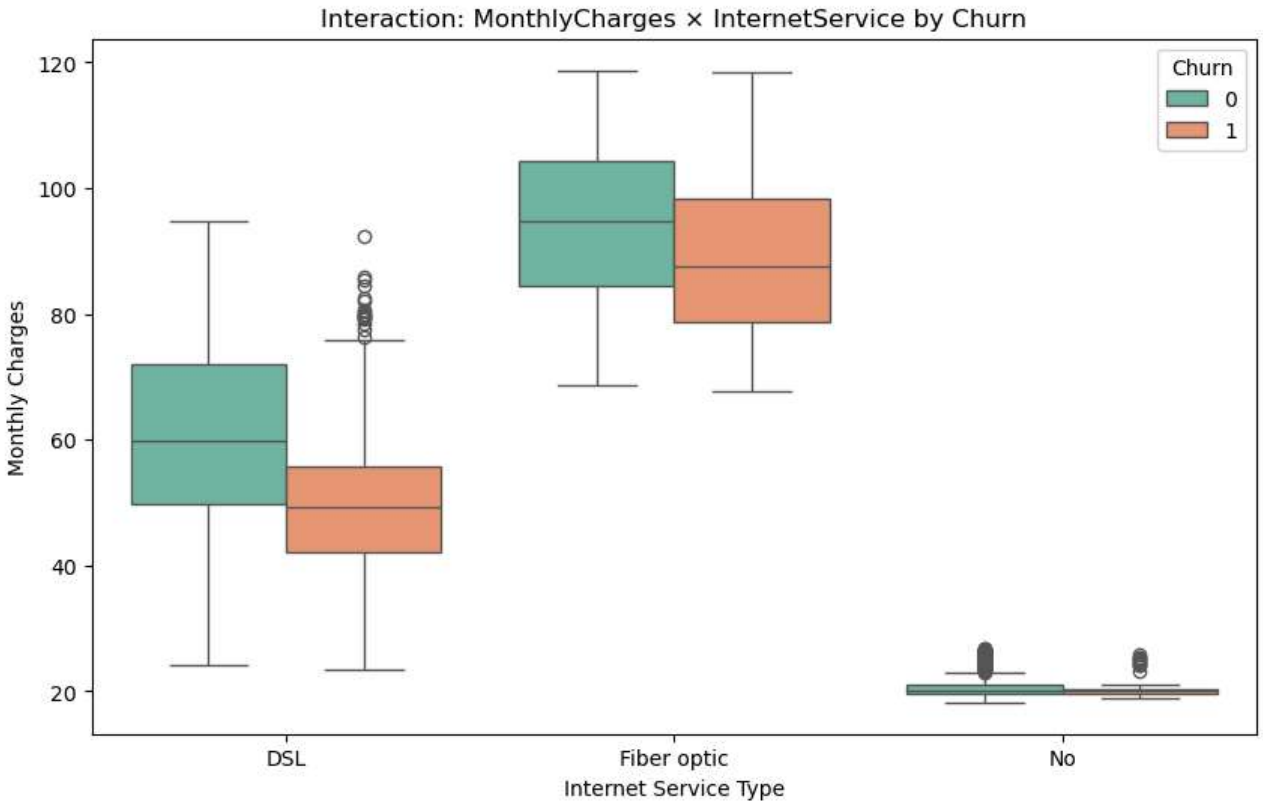
```
      tenure  MonthlyCharges  TotalCharges  Churn
0          1           29.85         29.85      0
1         34           56.95       1889.50      0
2          2           53.85        108.15      1
3         45           42.30       1840.75      0
4          2           70.70        151.65      1
...      ...             ...           ...    ...
7038      24           84.80       1990.50      0
7039      72          103.20       7362.90      0
7040      11           29.60        346.45      0
7041       4           74.40        306.60      1
7042      66          105.65       6844.50      0

[7043 rows x 4 columns]
                  tenure  MonthlyCharges  TotalCharges     Churn
tenure          1.000000        0.247900      0.825880 -0.352229
MonthlyCharges  0.247900        1.000000      0.651065  0.193356
TotalCharges    0.825880        0.651065      1.000000 -0.199484
Churn          -0.352229        0.193356     -0.199484  1.000000
```



In [154…

```python
#Interaction effects: e.g., MonthlyCharges × InternetService type on churn
plt.figure(figsize=(10, 6))

sns.boxplot(
    data=data,
    x='InternetService',
    y='MonthlyCharges',
    hue='Churn',
    palette='Set2'
```

```
)

plt.title('Interaction: MonthlyCharges × InternetService by Churn')
plt.ylabel('Monthly Charges')
plt.xlabel('Internet Service Type')
plt.legend(title='Churn')
plt.show()

#Compare how MonthlyCharges varies for churned vs. not churned customers within each Internet type (DSL, Fiber optic, No).
#higher churn with Fiber optic + high charges.
```



In [ ]:

In [155…

```
#High-risk churn customer segments profile.
#data['Churn'] = data['Churn'].map({'Yes': 1, 'No': 0})
data.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   object
 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      7032 non-null   float64
 20  Churn             7043 non-null   int64
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

In [156…

```python
data['Churn'] = pd.to_numeric(data['Churn'], errors='coerce')
print(data['Churn'].isna().sum())
```

```
0
```

In [157…

```python
churn_rate = data['Churn'].mean()
print(f"Overall Churn Rate: {churn_rate*100:.2f}%")
#~26-27% churn in this dataset
print("\nChurn by Contract:")
print(data.groupby('Contract')['Churn'].mean().sort_values(ascending=False))
#Month-to-month customers usually have the highest churn (≈43%),
#while Two year contracts have lowest churn (≈2–3%).

#Fiber optic customers churn much more than DSL users — likely due to pricing or service quality.
print("\nChurn by Internet Service:")
print(data.groupby('InternetService')['Churn'].mean().sort_values(ascending=False))

#Customers with <1 year tenure churn the most — new customers are more at risk.
bins = [0, 12, 24, 48, 72]
labels = ['<1 year', '1-2 years', '2-4 years', '4-6 years']
data['tenure_group'] = pd.cut(data['tenure'], bins=bins, labels=labels, include_lowest=True)
print(data.groupby('tenure_group')['Churn'].mean().sort_values(ascending=False))

#Customers paying by Electronic check are often the most likely to churn.
print(data.groupby('PaymentMethod')['Churn'].mean().sort_values(ascending=False))
#High monthly charges → higher churn risk.
data['MonthlyChargesRange'] = pd.cut(data['MonthlyCharges'], bins=[0, 35, 70, 120], labels=['Low', 'Medium', 'High'])
print(data.groupby('MonthlyChargesRange')['Churn'].mean().sort_values(ascending=False))
```

```
Overall Churn Rate: 26.54%

Churn by Contract:
Contract
Month-to-month    0.427097
One year          0.112695
Two year          0.028319
Name: Churn, dtype: float64

Churn by Internet Service:
InternetService
Fiber optic    0.418928
DSL            0.189591
No             0.074050
Name: Churn, dtype: float64
tenure_group
<1 year      0.474382
1-2 years    0.287109
2-4 years    0.203890
4-6 years    0.095132
Name: Churn, dtype: float64
PaymentMethod
Electronic check             0.452854
Mailed check                 0.191067
Bank transfer (automatic)    0.167098
Credit card (automatic)      0.152431
Name: Churn, dtype: float64
MonthlyChargesRange
High      0.353614
Medium    0.239420
Low       0.108934
Name: Churn, dtype: float64
```

```
C:\Users\kalia\AppData\Local\Temp\ipykernel_4204\13899394.py:17: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  print(data.groupby('tenure_group')['Churn'].mean().sort_values(ascending=False))
C:\Users\kalia\AppData\Local\Temp\ipykernel_4204\13899394.py:23: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  print(data.groupby('MonthlyChargesRange')['Churn'].mean().sort_values(ascending=False))
```

In [158…

```
# | Segment                       | Typical Behavior                                | Churn Rate          |
# | ----------------------------- | ----------------------------------------------- | ------------------- |
# | **Contract: Month-to-month**  | Short-term, less commitment                     | Very High        |
# | **Internet: Fiber optic**     | High-speed users, often price-sensitive         | High             |
# | **Payment: Electronic check** | Possibly less engaged, prefer quick cancellation | High             |
# | **Tenure: <1 year**           | New customers, low loyalty                      | Highest          |
# | **High MonthlyCharges**       | Expensive plans, potential dissatisfaction      | High             |
# | **Senior Citizens (optional)** | May find services complex or costly             | Moderate to High |
# High-risk churn customers are typically:
# On month-to-month contracts
# Using fiber optic internet
# Paying via electronic check
# Having tenure under 1 year
# Paying high monthly charges
# Often without online security or tech support add-ons
```

In [159…

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 23 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   customerID           7043 non-null   object
 1   gender               7043 non-null   object
 2   SeniorCitizen        7043 non-null   object
 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         7032 non-null   float64
 20  Churn                7043 non-null   int64
 21  tenure_group         7043 non-null   category
 22  MonthlyChargesRange  7043 non-null   category
dtypes: category(2), float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

In [160…

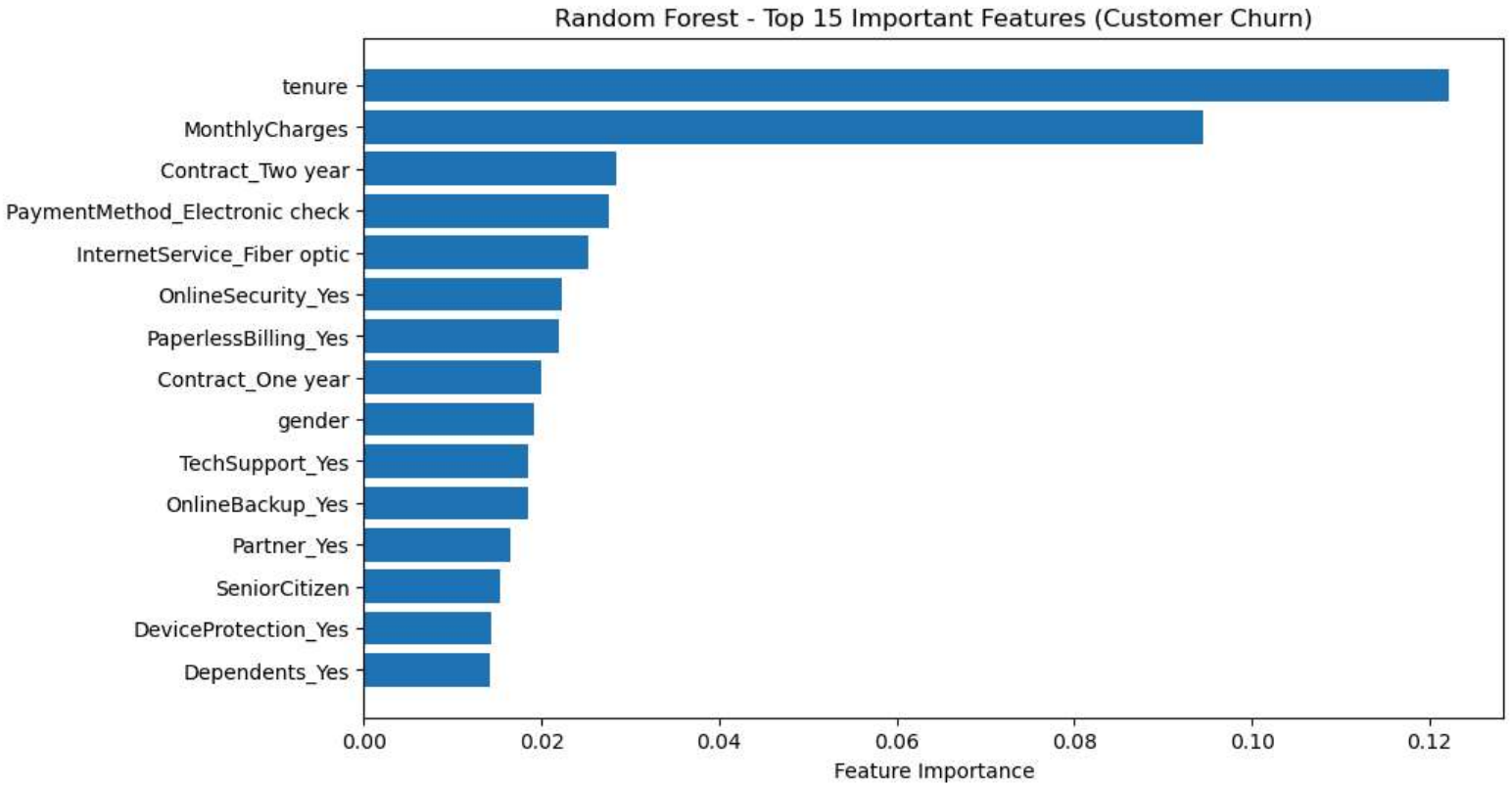
```python
# Load the raw dataset
data = pd.read_csv("Teleco_Customer_Churn.csv")

# Inspect the Churn column
print("Unique values before cleaning:", data['Churn'].unique())
# Keep only valid churn entries ('Yes'/'No')
data = data[data['Churn'].isin(['Yes', 'No'])]

# Encode target variable
data['Churn'] = data['Churn'].map({'Yes': 1, 'No': 0})

# Encode gender
data['gender'] = LabelEncoder().fit_transform(data['gender'])

# Drop customerID
X = data.drop(columns=['customerID', 'Churn'])
y = data['Churn']

# One-hot encode categorical columns
X = pd.get_dummies(X, drop_first=True)

# Split into train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print("\nDataset split completed successfully!")
print(f"Training samples: {X_train.shape[0]}")
print(f"Testing samples: {X_test.shape[0]}")
print(f"Churn ratio: {y.mean():.2%}")
```

```
Unique values before cleaning: ['No' 'Yes']

Dataset split completed successfully!
Training samples: 5634
Testing samples: 1409
Churn ratio: 26.54%
```

In [161…

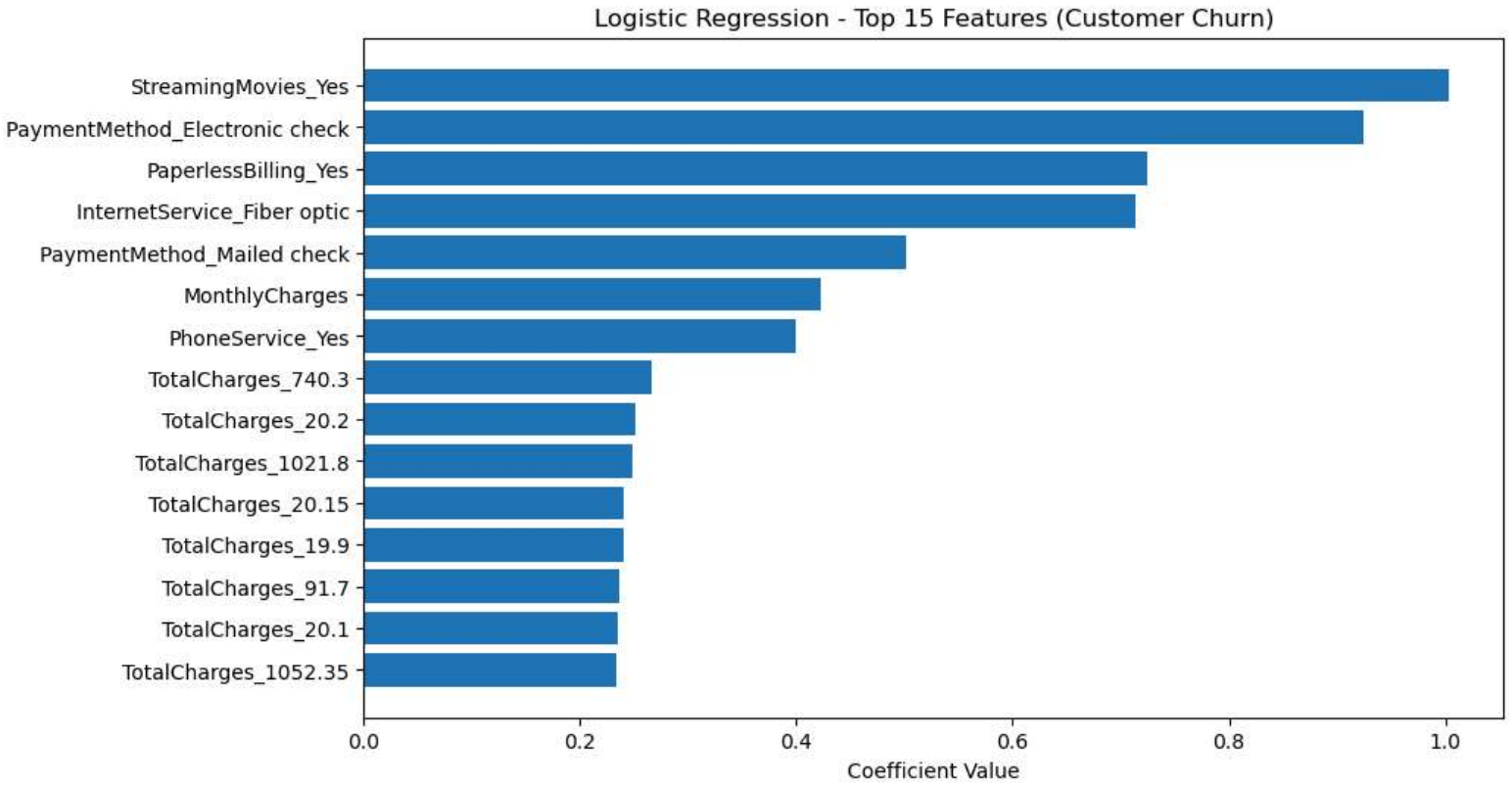
```python
# Train Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

# Make predictions
y_pred = rf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f"Random Forest Accuracy: {accuracy * 100:.2f}%")

# Get feature importance
rf_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf.feature_importances_
}).sort_values(by='Importance', ascending=False)

# Plot
plt.figure(figsize=(10, 6))
plt.barh(rf_importance['Feature'][:15], rf_importance['Importance'][:15])
plt.xlabel('Feature Importance')
plt.title('Random Forest - Top 15 Important Features (Customer Churn)')
plt.gca().invert_yaxis()
plt.show()
```

```
Random Forest Accuracy: 79.99%
```



In [162…

```python
# Scale features (important for Logistic Regression)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train Logistic Regression
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train_scaled, y_train)

# Get feature coefficients
lr_importance = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': lr.coef_[0]
}).sort_values(by='Coefficient', ascending=False)

# Plot
plt.figure(figsize=(10, 6))
plt.barh(lr_importance['Feature'][:15], lr_importance['Coefficient'][:15])
plt.xlabel('Coefficient Value')
plt.title('Logistic Regression - Top 15 Features (Customer Churn)')
plt.gca().invert_yaxis()
plt.show()
```



In [163…

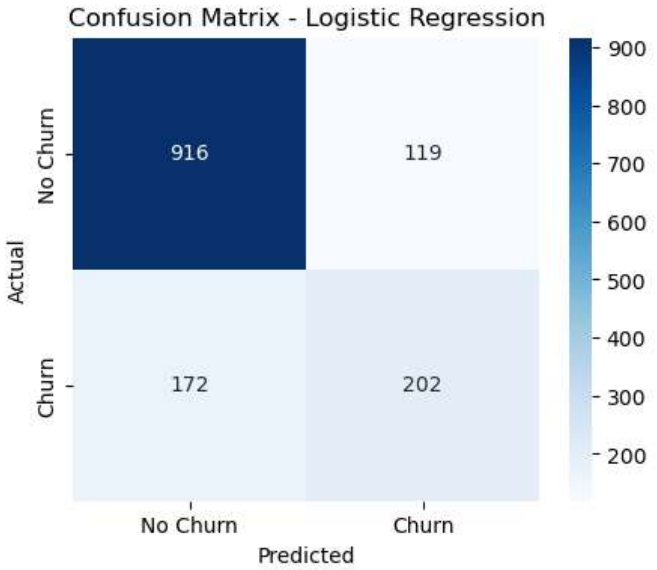
```
top_rf = rf_importance.head(10)
top_lr = lr_importance.head(10)

print("Top Random Forest Features:\n", top_rf)
print("Top Logistic Regression Features:\n", top_lr)
```

```
Top Random Forest Features:
                          Feature  Importance
2                          tenure    0.122216
3                  MonthlyCharges    0.094510
24              Contract_Two year    0.028415
27  PaymentMethod_Electronic check    0.027663
9      InternetService_Fiber optic    0.025230
12             OnlineSecurity_Yes    0.022367
25            PaperlessBilling_Yes    0.022026
23              Contract_One year    0.019967
0                          gender    0.019181
18                TechSupport_Yes    0.018548
Top Logistic Regression Features:
                            Feature  Coefficient
22              StreamingMovies_Yes     1.003931
27    PaymentMethod_Electronic check     0.925098
25              PaperlessBilling_Yes     0.724422
9         InternetService_Fiber optic     0.713462
28         PaymentMethod_Mailed check     0.500785
3                    MonthlyCharges     0.423186
6                   PhoneService_Yes     0.399536
5755              TotalCharges_740.3     0.266454
1638               TotalCharges_20.2     0.250553
71              TotalCharges_1021.8     0.247891
```

In [164…

```
#Predict churn using Logistic Regression; evaluate with confusion matrix
#Train Logistic Regression model
log_reg = LogisticRegression(max_iter=1000, random_state=42)
log_reg.fit(X_train, y_train)

#Make predictions
y_pred = log_reg.predict(X_test)

#Evaluate performance
print(" Model Evaluation:")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Churn', 'Churn'], yticklabels=['No Churn', 'Churn'])
plt.title('Confusion Matrix - Logistic Regression')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
 Model Evaluation:
Accuracy: 0.7935

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.89      0.86      1035
           1       0.63      0.54      0.58       374

    accuracy                           0.79      1409
   macro avg       0.74      0.71      0.72      1409
weighted avg       0.79      0.79      0.79      1409
```



In [165…

```
# Key Findings:

# Top predictors of churn:
```

```
# Tenure (shorter = higher churn risk)

# MonthlyCharges (higher = higher churn risk)

# Contract type (month-to-month contracts show higher churn)

# Payment method (electronic check customers churn more)

# Lack of Online Security or Tech Support services increases churn risk
```