

EDA- Telecom Customer Churn

A comprehensive exploration of customer data to identify churn patterns, analyze feature relationships, and uncover actionable business insights using Python.

Tools & Libraries Used:

Programming Language: Python

Libraries: Pandas, NumPy, Matplotlib, Seaborn

Environment: Jupyter Notebook

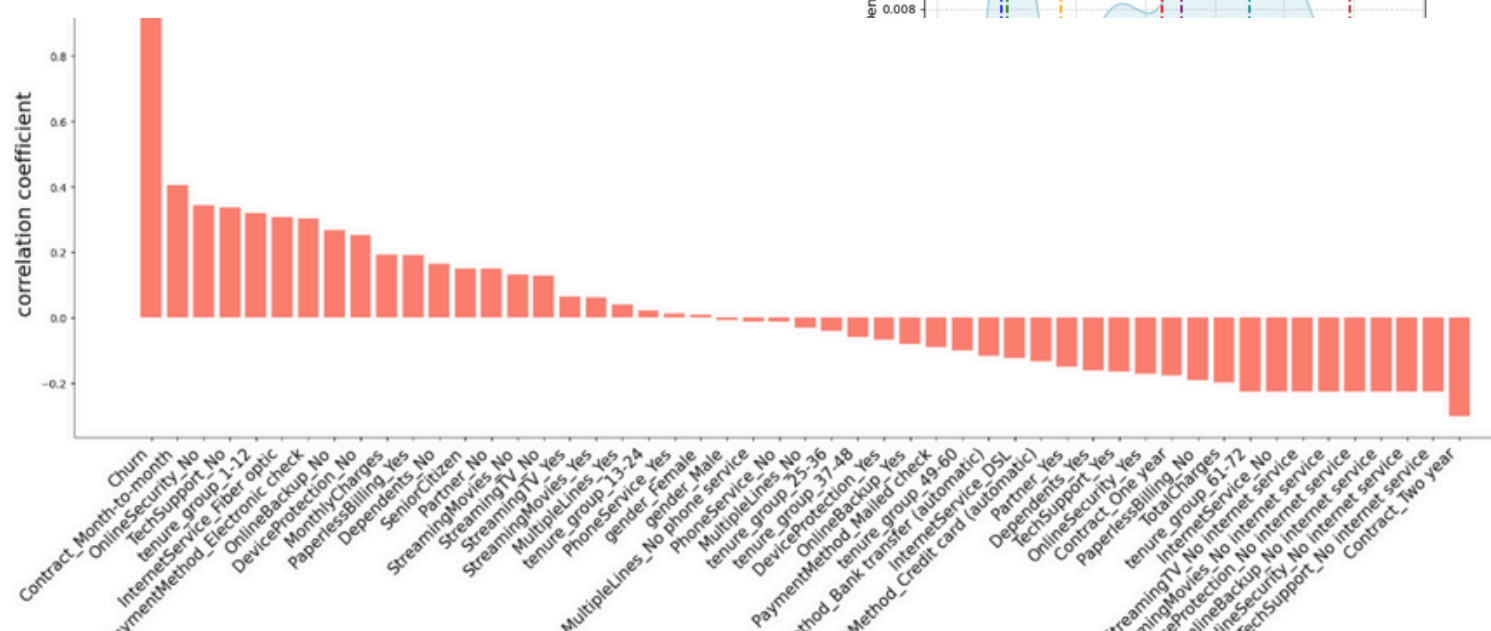
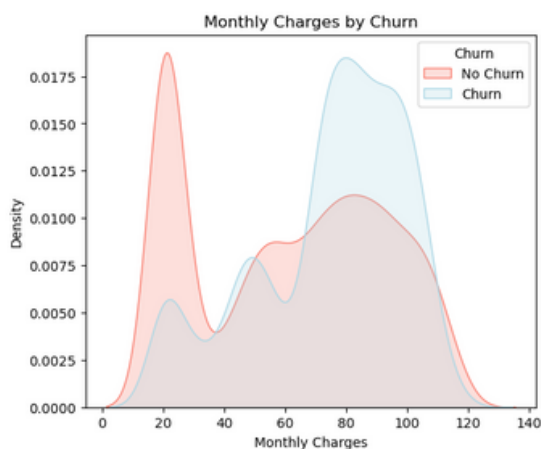
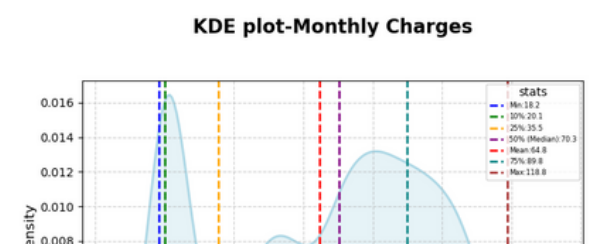
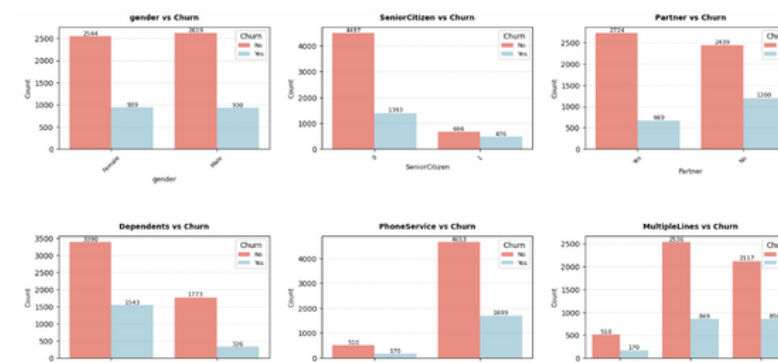
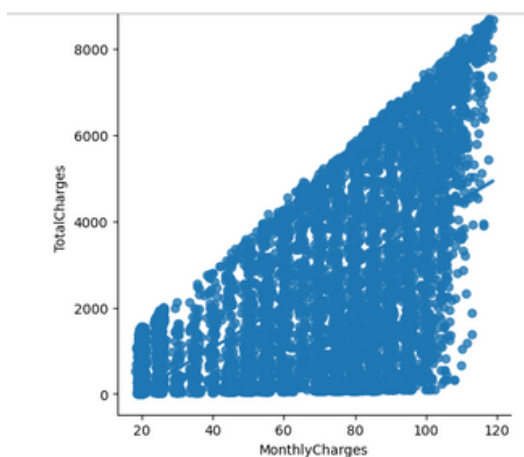
Visualization Type: KDE plots, Scatter plots, Correlation Bar charts

Dataset Used: Telco Customer Churn Dataset

Prepared By: Sankeerthana

Date: 11-November 2025

Peak & Spread View of Insights



Key Insights and Findings from the EDA

1. Data Overview

- **Data Shape:** 7043 x 21
- **Dataset:** Cleaned Telco Customer Churn data
- **Target Variable:** Churn (Yes = 1, No = 0)
- **Data Preparation Steps:**
 - Converted categorical Churn to binary format using `np.where`
 - Created dummy variables for categorical features using `pd.get_dummies()`
 - Ensured data types are consistent and removed null values

2. Key Descriptive Insights

- **75% of customers** have **tenure less than 55 months**, indicating a short average service duration.
- **Average Monthly Charges:** \$64.7
 - **Top 25% customers** pay more than **\$89.8 per month**.
- **Overall Churn Rate:** ~26%
- **Gender-based Churn:**
 - Male churn rate ≈ 26%
 - Female churn rate ≈ 26%
- **Senior Citizens:** 41% churn rate — much higher than non-seniors (23%)
- **Partner Status:**
 - No partner → **32% churn**
 - With partner → **19% churn**
- **Dependents:**
 - No dependents → **31% churn**
 - With dependents → **15% churn**

3. Service & Contract-Based Insights

- **Multiple Lines:** Slightly higher churn (~29%) — may indicate higher plan complexity or cost.
- **Internet Service:**
 - **Fiber Optic users:** Highest churn (~42%) — possibly due to premium pricing.
 - **DSL users:** More stable.
 - **No internet:** Very low churn (7%).
- **Online Security:**

Customers without security churn nearly **3x more** → offering security packages could improve retention.
- **Device Protection & Tech Support:**
 - Customers with these services churn less (~22%).
 - Tech support availability is a strong retention factor.
- **Streaming Services:**
 - Customers not streaming movies churn slightly more (34%) than those who do (30%) — engagement may reduce churn.
- **Contract Type:**
 - **Month-to-month:** Highest churn (43%)
 - **One-year:** Moderate churn (~11%)
 - **Two-year:** Lowest churn (3%) → longer commitments ensure higher retention.
- **Billing Type:**
 - **Paperless billing:** 34% churn
 - **Paper billing:** 16% churn → traditional users tend to stay longer.
- **Payment Method:**
 - **Electronic check users:** Highest churn (~45%)
 - **Automatic payments (bank transfer/credit card):** Lowest churn (~15–17%)
- **Tenure Group:**
 - **New customers (1–12 months)** are **8x more likely** to churn than **long-term customers (61–72 months)**.

4. Numerical Analysis Insights

KDE Plot – Monthly Charges

- The distribution is **bimodal** with two major peaks:
 - Around **\$20–25** (low-paying group)
 - Around **\$70–90** (high-paying group)
- **Mean:** \$64.8, **Median:** \$70.3 → slight right skew.
- **75% of customers** have Monthly Charges below **\$90.8**.
- A few pay up to **\$118.8** → premium plans or multiple services.
- Helps identify **customer segments**: low, mid, and high spenders.

Scatter Plot – Monthly Charges vs Total Charges

- Strong **positive correlation** between Monthly and Total Charges.
- The **triangular pattern** shows Total Charges depend on **tenure** — newer customers with high charges have lower totals.
- Confirms that **revenue grows with customer tenure**.

KDE Plot – Monthly Charges by Churn

- **Churn probability increases** with higher Monthly Charges.
- Non-churn customers cluster around **lower charge ranges**.
- Indicates **price sensitivity** as a churn driver.
- **Business Note:** Consider targeted loyalty offers for high-paying customers.

KDE Plot – Total Charges by Churn

- **Low Total Charges (0–2000)** customers show the **highest churn**.
- **Higher Total Charges** → **more loyal** customers (longer tenure).
- Emphasizes importance of **early customer engagement**.

Correlation Bar Plot

- **Top positive correlation with churn:**
 - Month-to-month contract
 - High monthly charges
 - Electronic check payment
- **Top negative correlation with churn:**
 - Longer tenure
 - Tech support, Online security, Two-year contracts
- Suggests that **service engagement and longer commitment terms** drive retention.

Business Recommendations

- **Improve Early Retention:**
 - Focus on onboarding and customer support in the first 6-12 months, where churn risk is highest.
- **Encourage Long-Term Contracts:**
 - Offer discounts or perks for switching from month-to-month to annual plans.
- **Promote Automatic Payments:**
 - Provide incentives for switching to auto-payment methods — customers using them churn less.
- **Bundle Support & Security Services:**
 - Promote tech support and online security packages as value-added retention features.
- **Target High-Charge Customers:**
 - Introduce loyalty offers or pricing flexibility for customers with high monthly charges to prevent dissatisfaction.
- **Engagement Campaigns for At-Risk Segments:**
 - Senior citizens and customers without dependents or partners need special retention strategies (personalized communication, offers).

```
[1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
%matplotlib inline
```

```
[2]: telco_base_data=pd.read_csv('CustomerChurn.csv')
```

```
[3]: telco_base_data.head()
```

[3]:	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	St
	0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No
	1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No
	2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No
	3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes

	2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No
	3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes
	4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No

5 rows × 21 columns



```
[5]: telco_base_data.shape
```

```
[5]: (7043, 21)
```

```
[6]: telco_base_data.columns.values
```

```
[6]: array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
'TotalCharges', 'Churn'], dtype=object)
```

```
[7]: telco_base_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
```

```
[8]: telco_base_data.describe()
```

```
[8]:
```

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

```
[19]: # Object Oriented Visual
fig,ax=plt.subplots(figsize=(7,5))
sns.kdeplot(telco_base_data['MonthlyCharges'],ax=ax,color='lightblue',fill=True,alpha=0.3,linewidth=2)

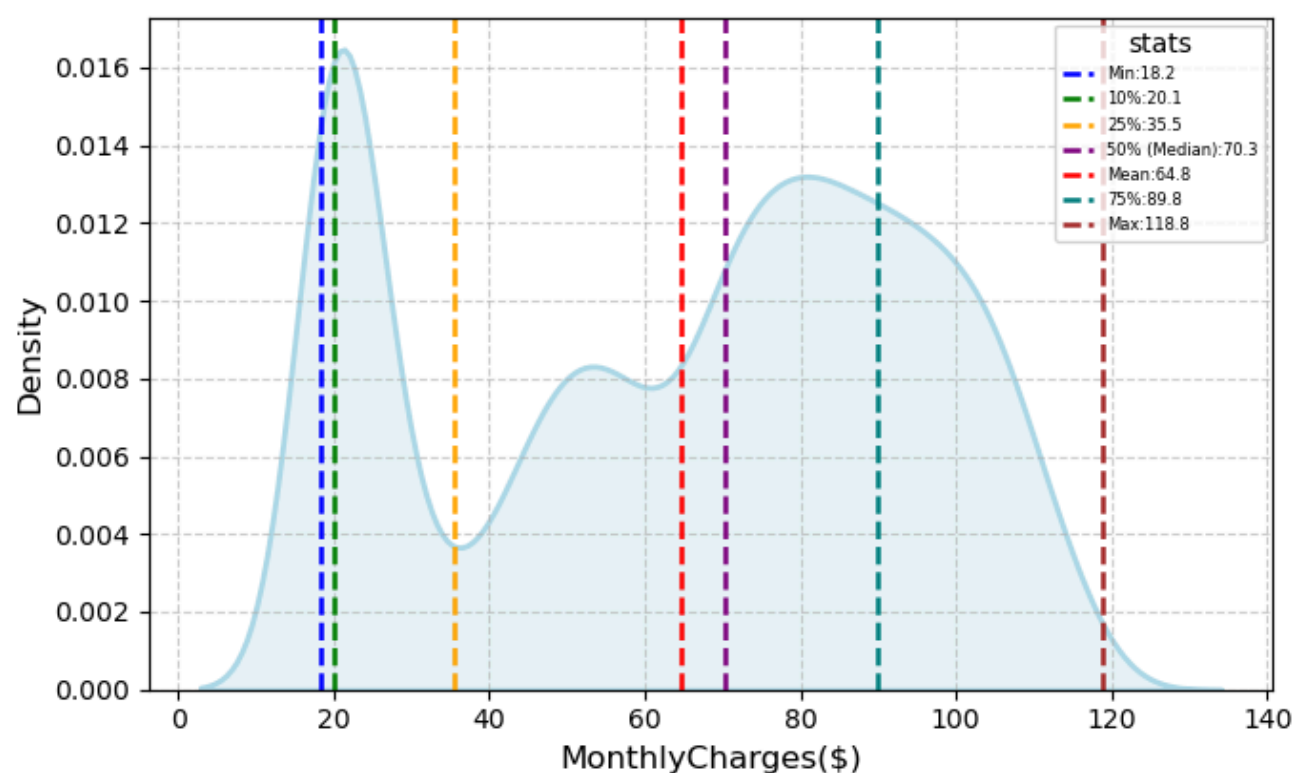
stats={
    'Min':('blue',telco_base_data['MonthlyCharges'].min()),
    '10%':('green',telco_base_data['MonthlyCharges'].quantile(0.10)),
    '25%':('orange',telco_base_data['MonthlyCharges'].quantile(0.25)),
    '50% (Median)':('purple',telco_base_data['MonthlyCharges'].median()),
    'Mean':('red',telco_base_data['MonthlyCharges'].mean()),
    '75%':('teal',telco_base_data['MonthlyCharges'].quantile(0.75)),
    'Max':('brown',telco_base_data['MonthlyCharges'].max())
}

for label,(color,val)in stats.items():
    ax.axvline(val,linestyle='--',color= color,linewidth=1.8,label=f'{label}:{val:.1f}')
ax.set_title('KDE plot-Monthly Charges',fontsize=16,weight='bold',pad=40)

ax.set_xlabel('MonthlyCharges($)',fontsize=12)
ax.set_ylabel('Density',fontsize=12)
ax.grid(True,linestyle='--',alpha=0.6)
ax.legend(loc='upper right',fontsize=6,title='stats')

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

KDE plot-Monthly Charges



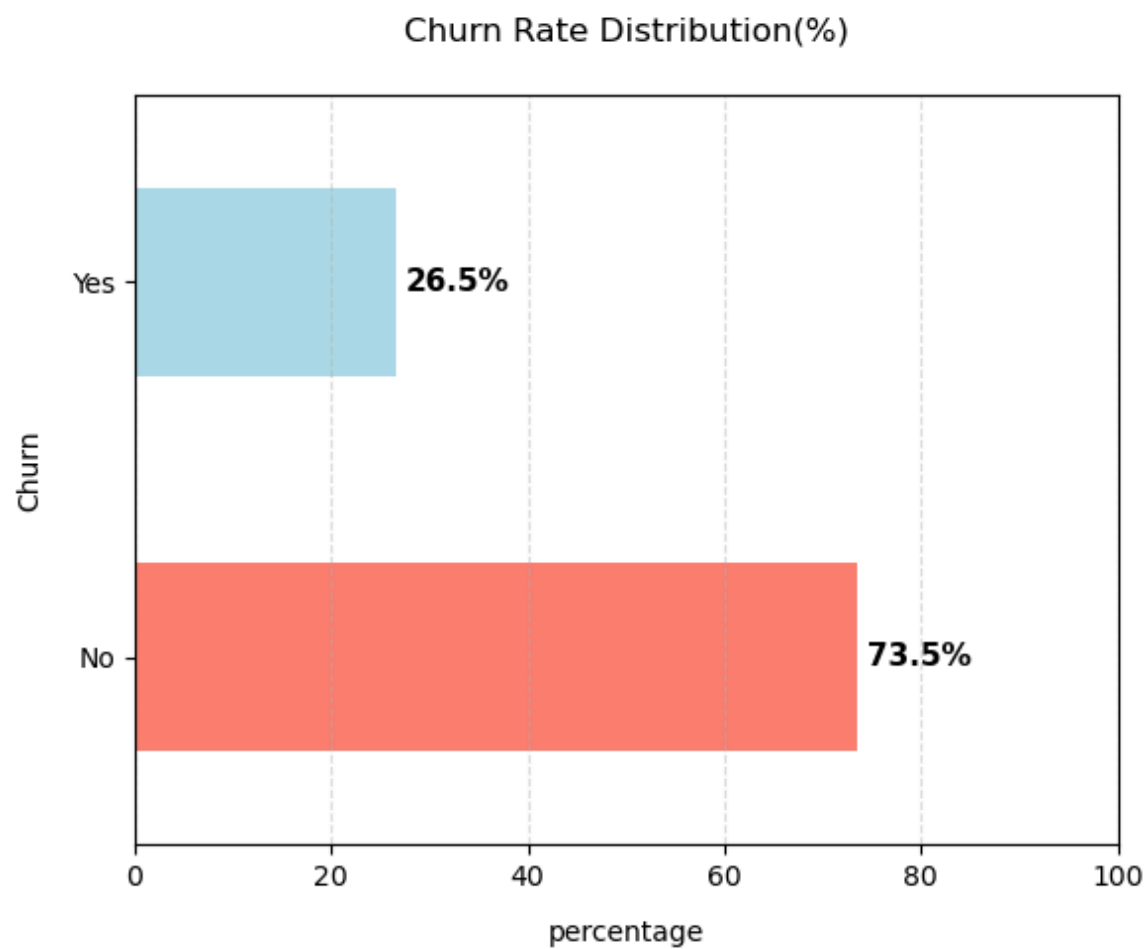
```
[20]: # % of churn count
telco_base_data['Churn'].value_counts()*100/len(telco_base_data['Churn'])
```

```
[20]: Churn
No    73.463013
Yes   26.536987
Name: count, dtype: float64
```

```
[21]: telco_base_data['Churn'].value_counts()
```

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

```
[27]: counts=telco_base_data['Churn'].value_counts()
percentages=counts/counts.sum()*100
percentages.plot(kind='barh',color=['salmon','lightblue'],figsize=(6,5))
for i,v in enumerate(percentages):
    plt.text(v+1,i,f'{v:.1f}%',va='center',fontsize=11,fontweight='bold')
plt.xlabel('percentage',labelpad=10)
plt.ylabel('Churn',labelpad=10)
plt.title('Churn Rate Distribution(%)',pad=20)
plt.grid(axis='x',linestyle='--',alpha=0.4)
plt.xlim(0,100)
plt.tight_layout()
plt.show()
```



```
[28]: telco_base_data.info(verbose=True)
```

```
<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
```

```
[29]: telco_data=telco_base_data.copy()
```

```
[30]: telco_data.TotalCharges=pd.to_numeric(telco_data.TotalCharges,errors='coerce')
telco_data.isnull().sum()
```

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

```
[31]: telco_data.dropna(how='any',inplace=True)
telco_data.isnull().sum()
```

```
[31]: 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
```

```
[ ]: # Analysis
```

```
[32]: print(telco_data['tenure'].max())

72
```

```
[33]: #binning
labels=["{}-{}".format(i,i+11) for i in range(1,72,12)]
telco_data['tenure_group']=pd.cut(telco_data.tenure,range(1,80,12),right=False,labels=labels)
telco_data['tenure_group'].value_counts()
```

```
[33]: tenure_group
1-12      2175
61-72     1407
13-24     1024
25-36      832
49-60      832
37-48      762
Name: count, dtype: int64
```

```
[34]: telco_data.head(7)
```

[34]:	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	TechSupport	Str
	0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No
		5575-											

```
[74]: telco_data.head(3)
```

[74]:	ackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn	tenure_group
	Yes	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	0	1-12
	No	Yes	No	No	No	One year	No	Mailed check	56.95	1889.50	0	25-36
	Yes	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	1	1-12

```
[39]: # telco_data.drop(columns=['tenure'],axis=1,inplace=True)
```

```
[ ]: # perfoming Analysis
```

```
[49]: import matplotlib.pyplot as plt
import seaborn as sns

predictors = telco_data.drop(columns=['customerID', 'Churn', 'TotalCharges', 'MonthlyCharges']).columns

n_cols = 3
n_rows = (len(predictors) + n_cols - 1) // n_cols

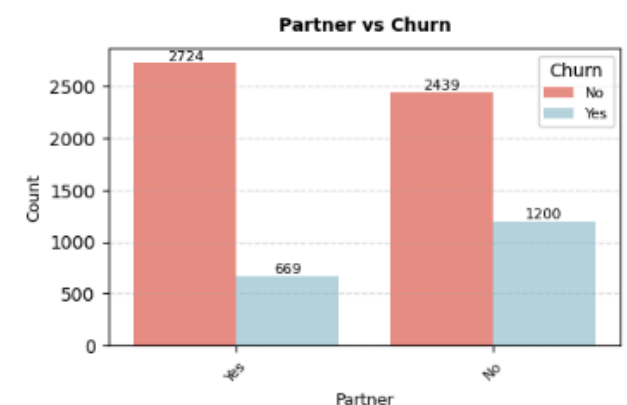
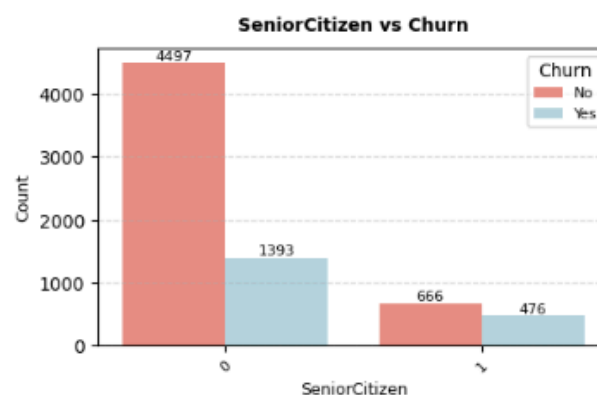
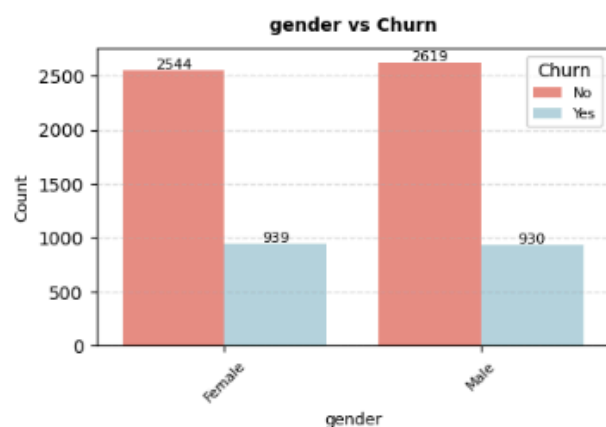
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, n_rows * 4))
axes = axes.flatten()

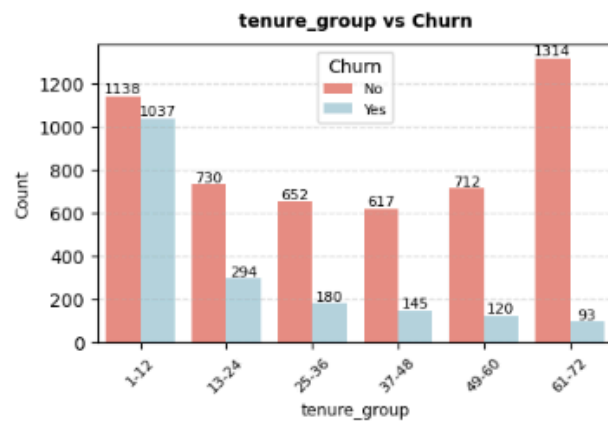
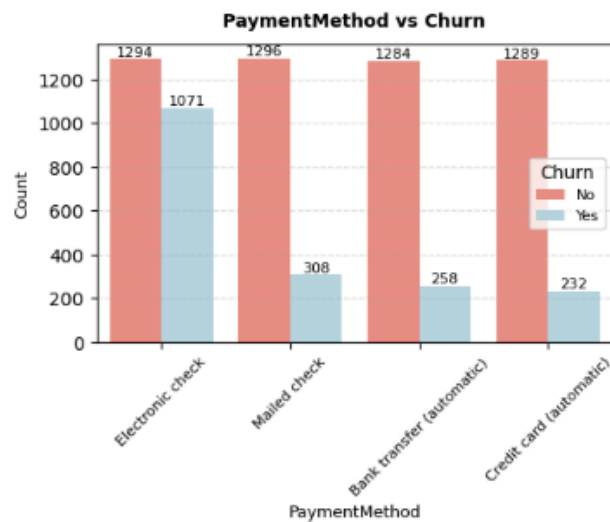
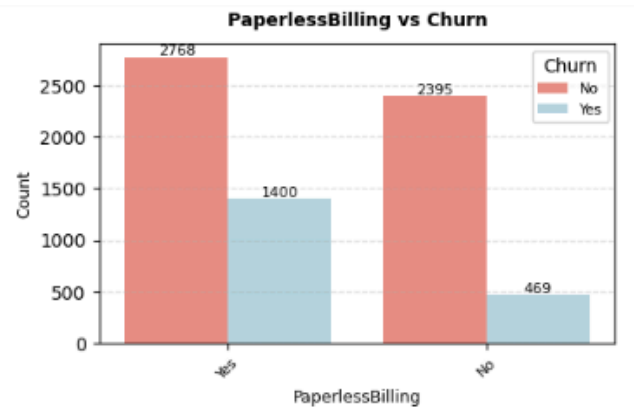
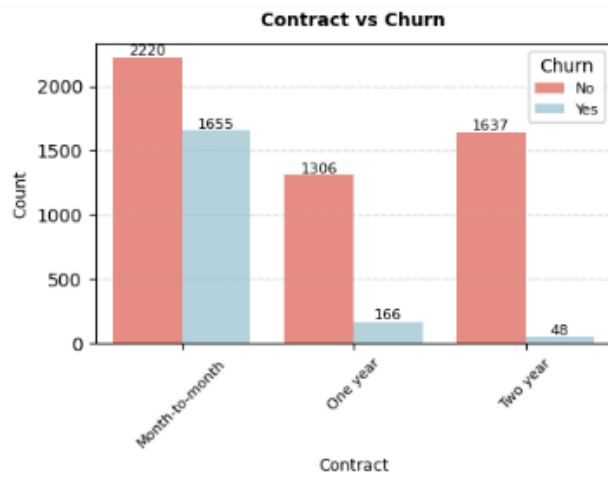
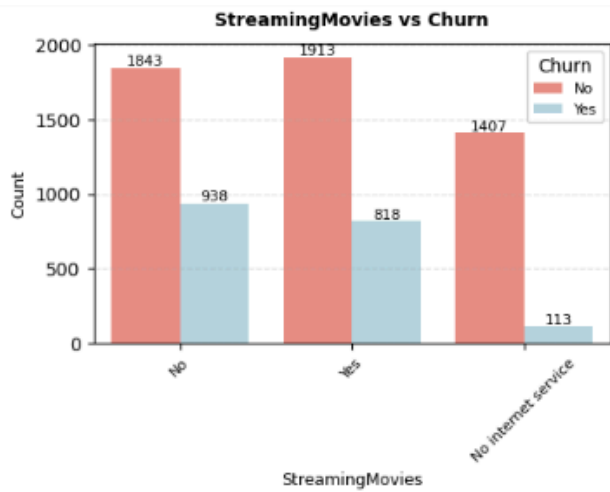
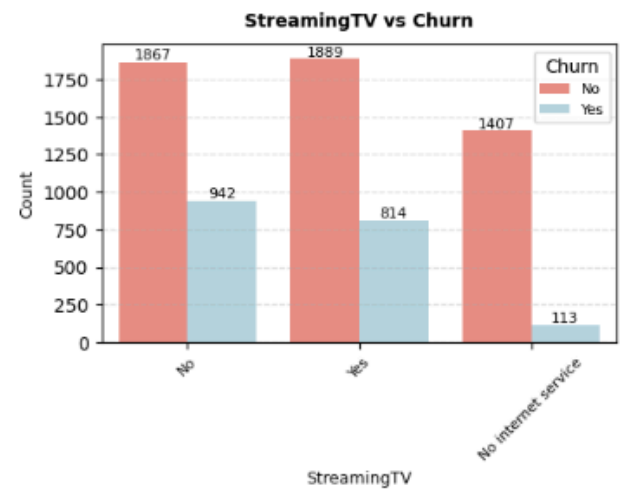
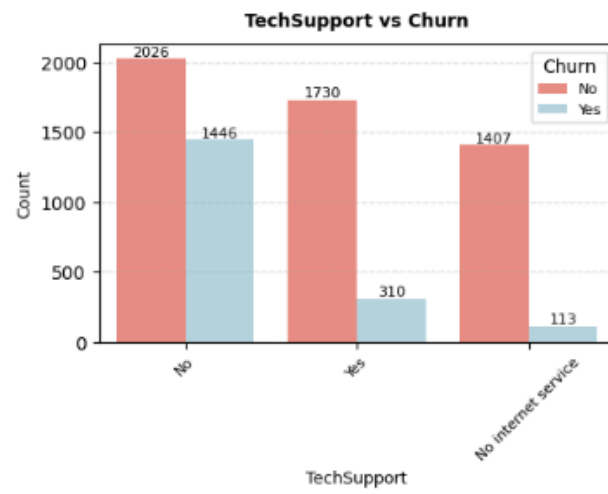
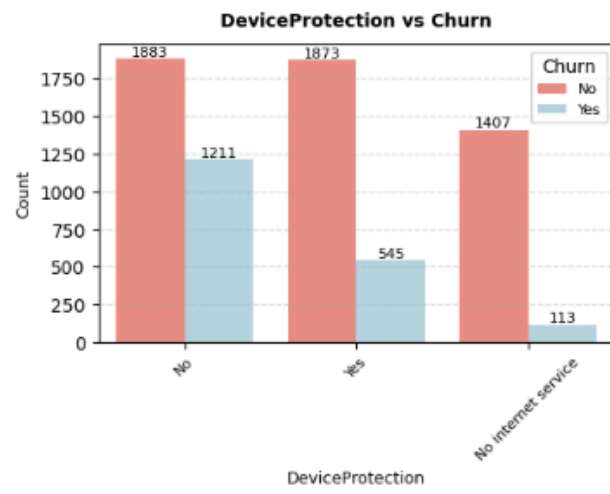
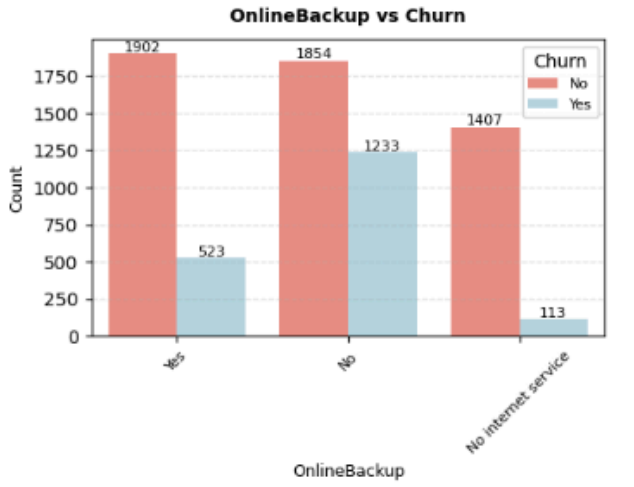
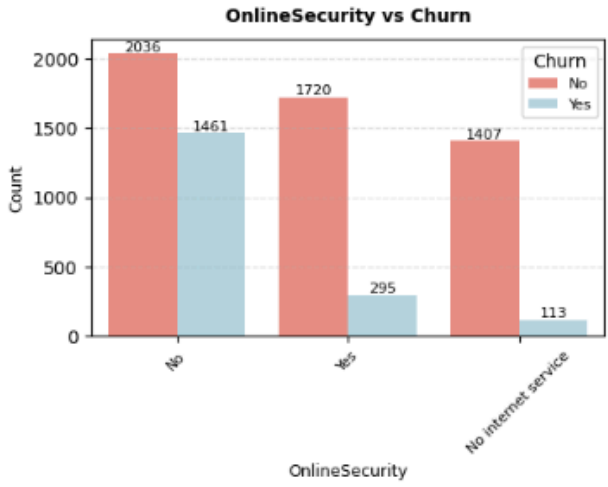
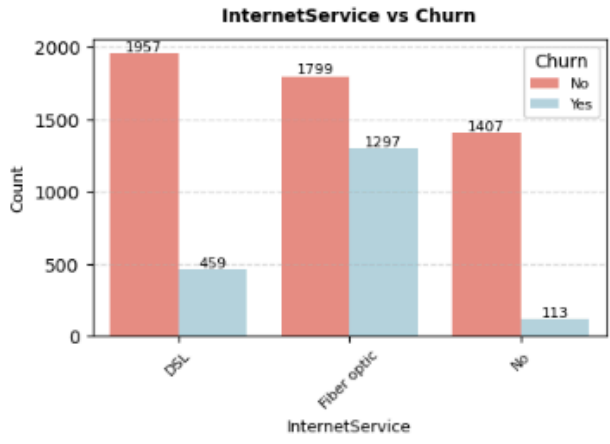
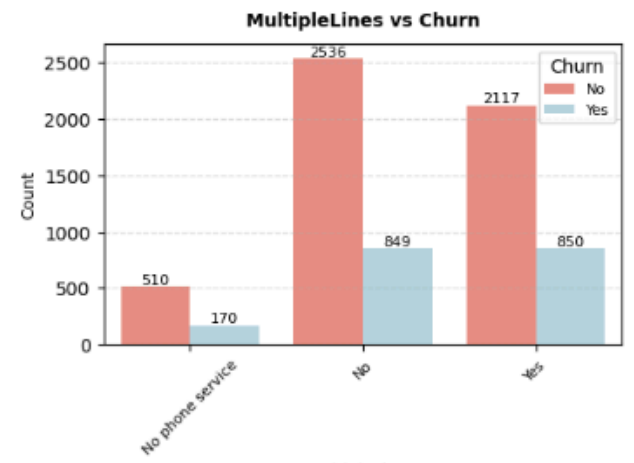
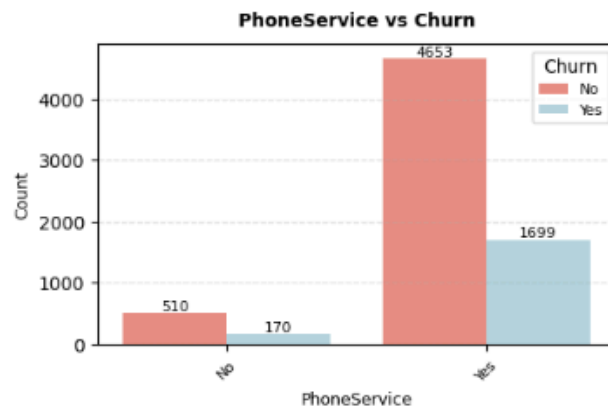
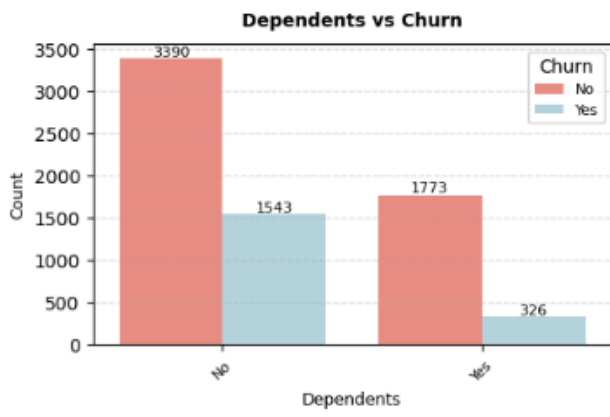
for i, predictor in enumerate(predictors):
    ax = axes[i]
    sns.countplot(data=telco_data, x=predictor, hue='Churn', ax=ax,
                  palette=['salmon', 'lightblue'])
    ax.set_title(f'{predictor} vs Churn', fontsize=10, weight='bold', pad=10)
    ax.set_xlabel(predictor, fontsize=9)
    ax.set_ylabel('Count', fontsize=9)
    ax.tick_params(axis='x', rotation=45, labels=8)
    ax.grid(axis='y', linestyle='--', alpha=0.4)
    ax.legend(title='Churn', fontsize=8)
```

```
# Add count labels above bars
for p in ax.patches:
    height = p.get_height()
    if height > 0:
        ax.annotate(f'{int(height)}',
                    (p.get_x() + p.get_width()/2., height),
                    ha='center', va='bottom', fontsize=8, color='black')

# Hide extra empty axes (if total plots < grid slots)
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout(pad=2)
plt.show()
```





[]: #Numerical Analysis

[50]: telco_data['Churn']=np.where(telco_data.Churn=='Yes',1,0)
telco_data.head()

[50]:	customerID	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	...	TechSupport	StreamingTV
0	7590-VHVEG	Female	0	Yes	No	No	No phone service	DSL	No	Yes	...	No	No
1	5575-GNVDE	Male	0	No	No	Yes	No	DSL	Yes	No	...	No	No
2	3668-QPYBK	Male	0	No	No	Yes	No	DSL	Yes	Yes	...	No	No
3	7795-CFOCW	Male	0	No	No	No	No phone service	DSL	Yes	No	...	Yes	No
4	9237-HQITU	Female	0	No	No	Yes	No	Fiber optic	No	No	...	No	No

5 rows × 21 columns

[]: telco_data.drop(columns=

[52]: telco_data.drop(columns=['customerID'],axis=1,inplace=True)
telco_data.head()

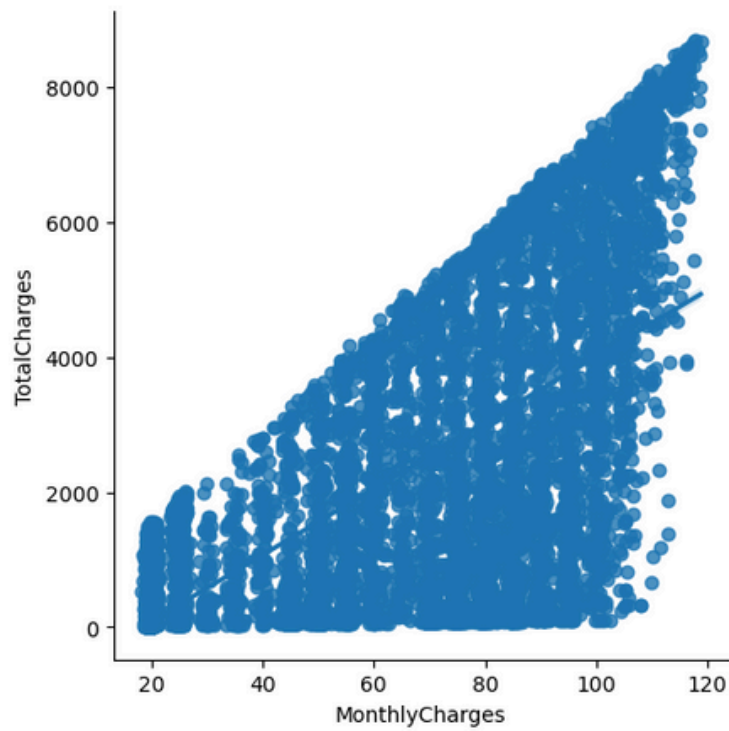
[52]:	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV
0	Female	0	Yes	No	No	No phone service	DSL	No	Yes	No	No	No
1	Male	0	No	No	Yes	No	DSL	Yes	No	Yes	No	No
2	Male	0	No	No	Yes	No	DSL	Yes	Yes	No	No	No
3	Male	0	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	No
4	Female	0	No	No	Yes	No	Fiber optic	No	No	No	No	No

[54]: telco_data_dummies=pd.get_dummies(telco_data)
telco_data_dummies.head()

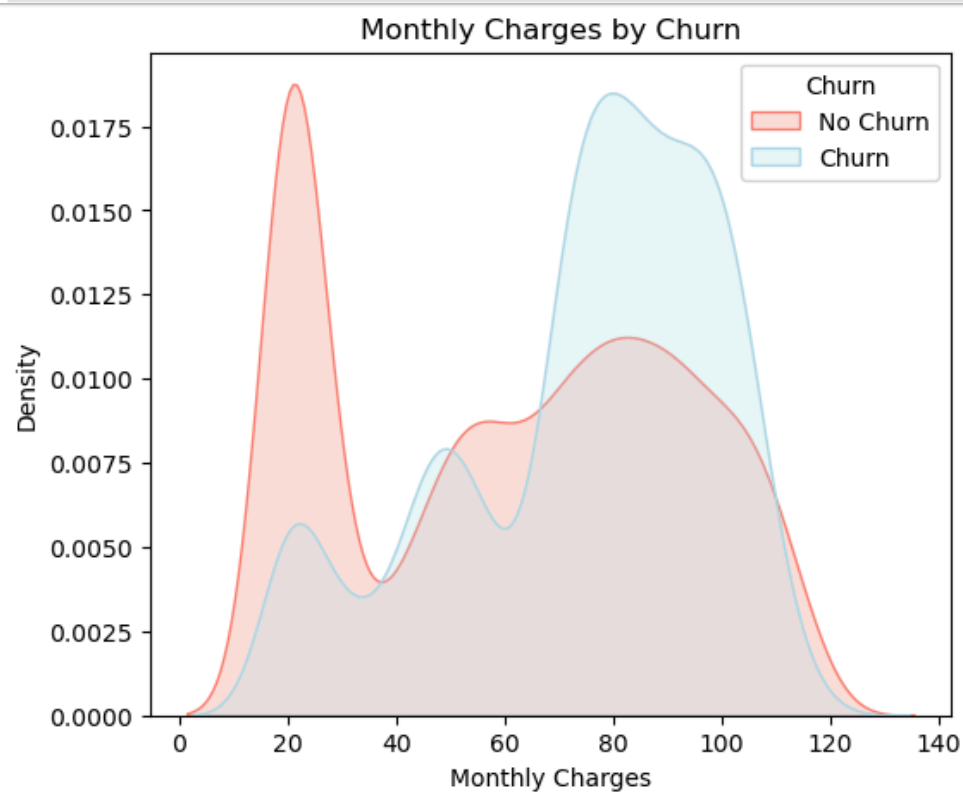
[54]:	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	...	PaymentMethod_Basic	PaymentMethod_Basic transfer (automatic)
0	0	29.85	29.85	0	True	False	False	True	True	False	...	False	False
1	0	56.95	1889.50	0	False	True	True	False	True	False	...	False	False
2	0	53.85	108.15	1	False	True	True	False	True	False	...	False	False
3	0	42.30	1840.75	0	False	True	True	False	True	False	...	False	True
4	0	70.70	151.65	1	True	False	True	False	True	False	...	False	False

5 rows × 51 columns

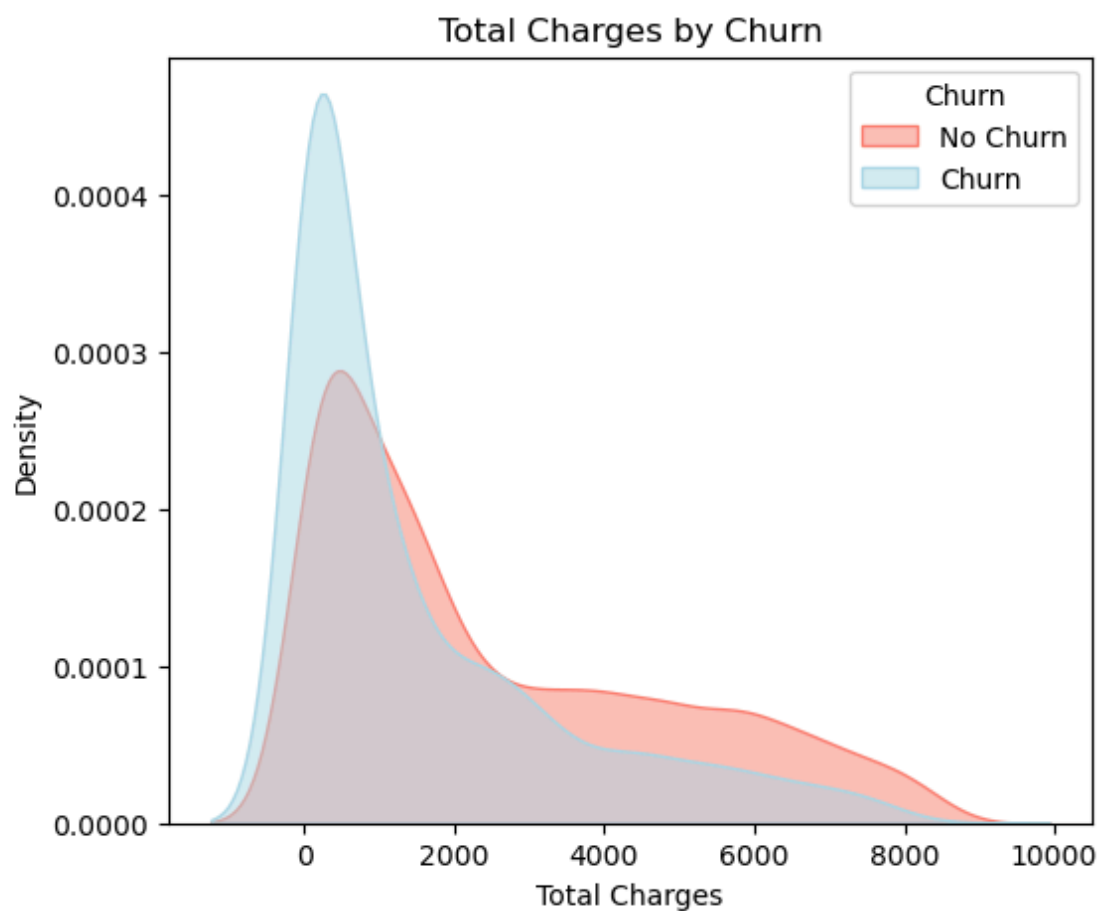
```
[58]: g=sns.lmplot(data=telco_data_dummies,x='MonthlyCharges',y='TotalCharges',fit_reg=True)
plt.show()
```



```
[66]: fig, ax = plt.subplots(figsize=(6, 5))
sns.kdeplot(
    data=telco_data_dummies[telco_data_dummies['Churn'] == 0],
    x='MonthlyCharges',
    color='salmon',
    fill=True,
    label='No Churn',
    ax=ax
)
sns.kdeplot(
    data=telco_data_dummies[telco_data_dummies['Churn'] == 1],
    x='MonthlyCharges',
    color='lightblue',
    fill=True,
    label='Churn',
    ax=ax
)
ax.set_xlabel('Monthly Charges')
ax.set_ylabel('Density')
ax.set_title('Monthly Charges by Churn')
ax.legend(title='Churn', loc='upper right')
plt.show()
```



```
[67]: fig, ax = plt.subplots(figsize=(6, 5))
sns.kdeplot(
    data=telco_data_dummies[telco_data_dummies['Churn'] == 0],
    x='TotalCharges',
    color='salmon',
    fill=True,
    label='No Churn',
    alpha=0.5,
    ax=ax
)
sns.kdeplot(
    data=telco_data_dummies[telco_data_dummies['Churn'] == 1],
    x='TotalCharges',
    color='lightblue',
    fill=True,
    label='Churn',
    alpha=0.5,
    ax=ax
)
ax.set_xlabel('Total Charges')
ax.set_ylabel('Density')
ax.set_title('Total Charges by Churn')
ax.legend(title='Churn', loc='upper right')
plt.show()
```



```
[73]: churn_corr=telco_data_dummies.corr(numeric_only=True)['Churn'].sort_values(ascending=False)
fig,ax=plt.subplots(figsize=(20,10))
ax.bar(churn_corr.index,churn_corr.values, color='salmon')
ax.set_title('Correlation of features with churn',fontsize=19)
ax.set_ylabel('correlation coefficient',fontsize=19)
ax.set_xticklabels(churn_corr.index,rotation=45,ha='right',fontsize=15)
plt.tight_layout()
plt.show()
```

C:\Users\SANKEERTHANA\AppData\Local\Temp\ipykernel_11012\1063636754.py:6: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
ax.set_xticklabels(churn_corr.index,rotation=45,ha='right',fontsize=15)

