```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

Generate | randomly select 5 items from a list | Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

```
df=pd.read_csv('/content/WA_Fn-UseC_-Telco-Customer-Churn.csv')
df
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneServ
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	

7043 rows × 21 columns

```
df.columns
```

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

```
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         object
Churn                object
dtype: object
```

NameError ValueError NameError •••

You

Please explain the error:

```
NameError: name 'pred' is not defined
```

Gemini

Hypothesis

The error message "NameError: name 'pred' is not defined" means that the variable `pred` you are trying to use in the `classification_report` and `confusion_matrix` functions hasn't been defined yet. Looking at your code, it seems you might have intended to use `predR` instead of `pred`.

Suggested Changes

```
print(classification_report(Y_test,predR)) # Use
print(confusion_matrix(Y_test,predR)) # Use pred
```

Use code with caution

Rate this answer

```
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
```

```
df.shape
```

```
(7043, 21)
```

```
df.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
```

```
df.describe()
```

	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

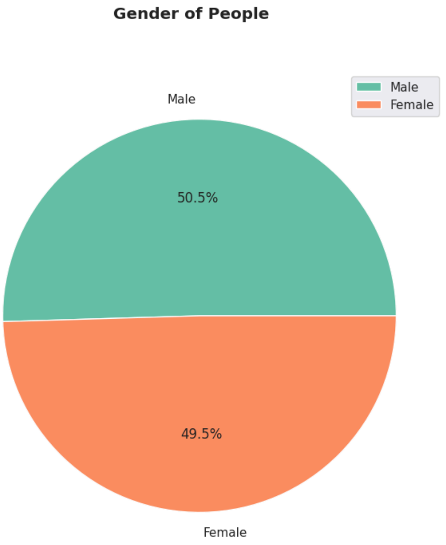
```
df.duplicated().sum()
```

```
0
```

```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'])
```

CATEGORICAL COLUMNS EXPLORATION

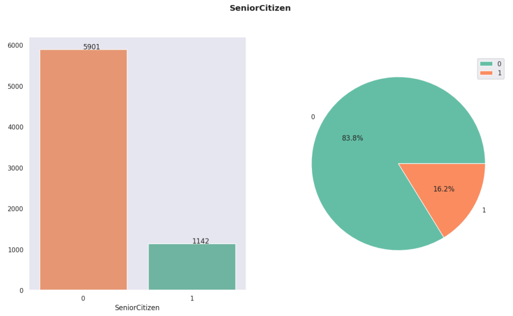
```
gender_counts = df['gender'].value_counts()
x = gender_counts.index
y = gender_counts.values

fig, ax = plt.subplots(figsize=(8, 8))

#pie chart
pal = sns.color_palette("Set2", len(gender_counts))
ax.pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Gender of People', weight='bold')
plt.show()
```



senior citizen

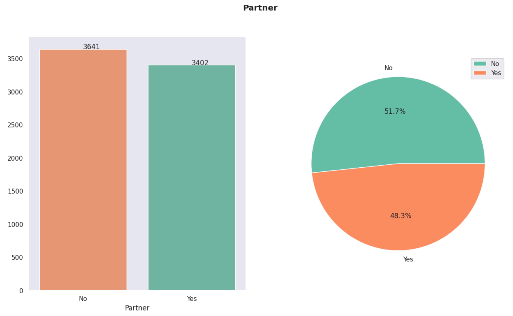
```
senior_counts = df['SeniorCitizen'].value_counts()
x = senior_counts.index
y = senior_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette( "Set2",len(senior_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('SeniorCitizen')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

# legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('SeniorCitizen', weight='bold')
plt.show()
```

```
<ipython-input-65-5f357ac7da10>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[6
```



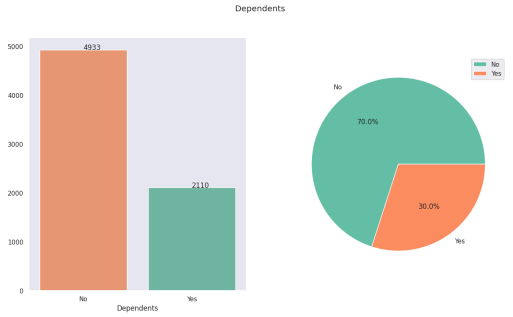
```
partner_counts = df['Partner'].value_counts()
x = partner_counts.index
y = partner_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(partner_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])
for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('Partner')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Partner', weight='bold')
plt.show()
```

```
<ipython-input-64-2a6a7b4bb7b4>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[(
```



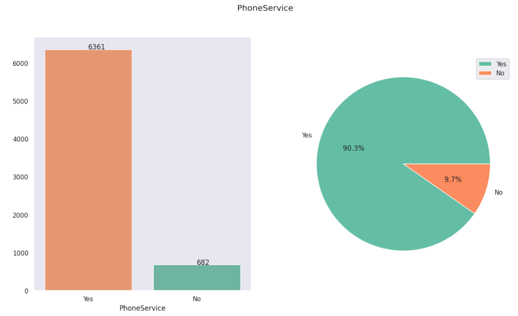
```
dependents_counts = df['Dependents'].value_counts()
x = dependents_counts.index
y = dependents_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(dependents_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('Dependents')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Dependents')
plt.show()
```

```
<ipython-input-63-909a902d337f>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[(
```



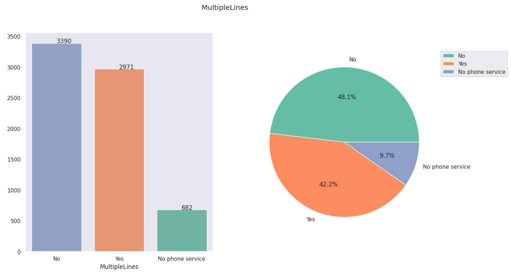
```
phone_service_counts = df['PhoneService'].value_counts()
x = phone_service_counts.index
y = phone_service_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(phone_service_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('PhoneService')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('PhoneService')
plt.show()
```

```
<ipython-input-62-55742b8ebd07>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0
```



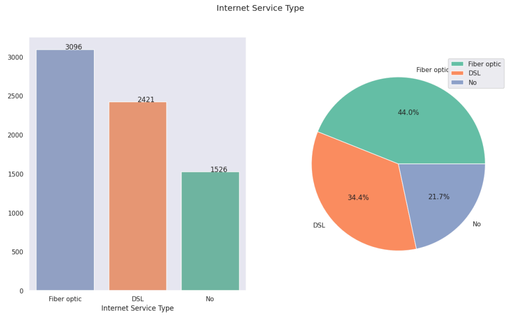
```
multiple_lines_counts = df['MultipleLines'].value_counts()
x = multiple_lines_counts.index
y = multiple_lines_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(multiple_lines_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('MultipleLines')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('MultipleLines')
plt.show()
```

```
<ipython-input-61-dac617941059>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[
```



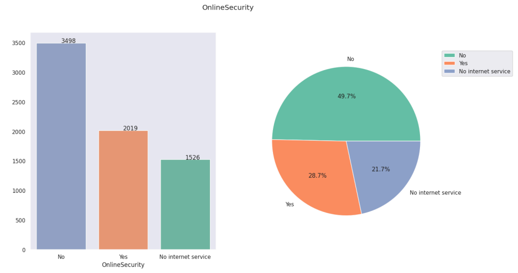
```
internet_service_counts = df['InternetService'].value_counts()
x = internet_service_counts.index
y = internet_service_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(internet_service_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('Internet Service Type')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Internet Service Type')
plt.show()
```

```
<ipython-input-60-c247801fd22e>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[
```



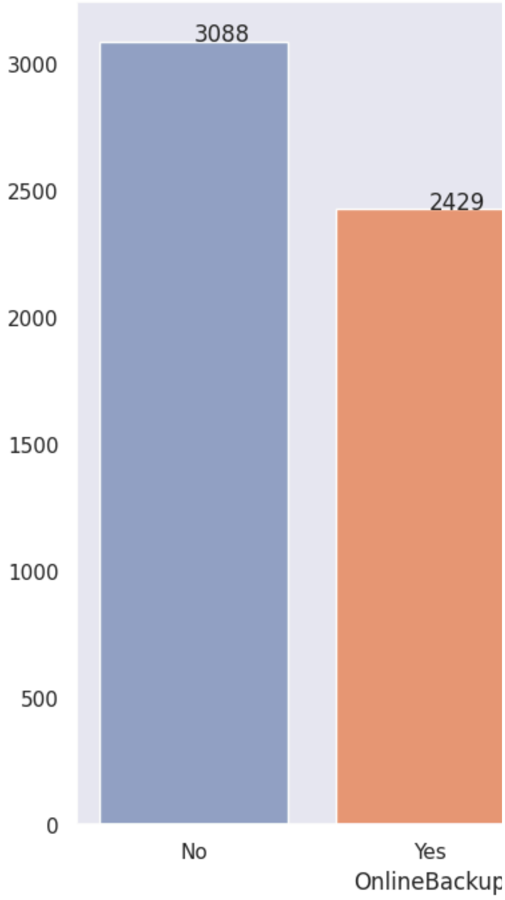
```
online_security_counts = df['OnlineSecurity'].value_counts()
x = online_security_counts.index
y = online_security_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(online_security_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('OnlineSecurity')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('OnlineSecurity')
plt.show()
```

```
<ipython-input-59-c977e446e677>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[
```



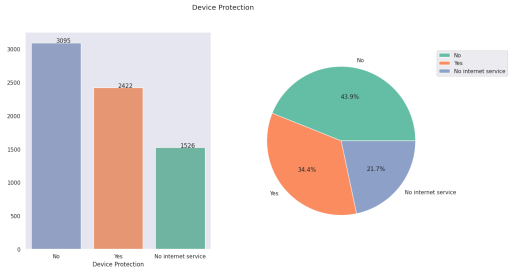
```
online_backup_counts = df['OnlineBackup'].value_counts()
x = online_backup_counts.index
y = online_backup_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))


#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(online_backup_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('OnlineBackup')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('OnlineBackup')
plt.show()
```

```
<ipython-input-58-466e6472fb68>:11: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[
```



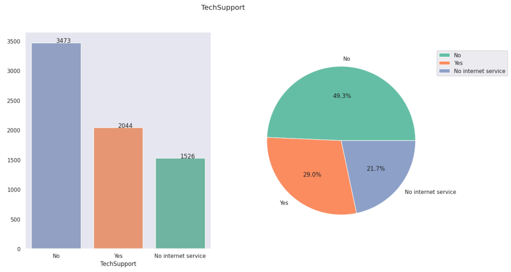
```
device_protection_counts = df['DeviceProtection'].value_counts()
x = device_protection_counts.index
y = device_protection_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(device_protection_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('Device Protection')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Device Protection')
plt.show()
```

```
<ipython-input-77-045e4563a15b>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0
```



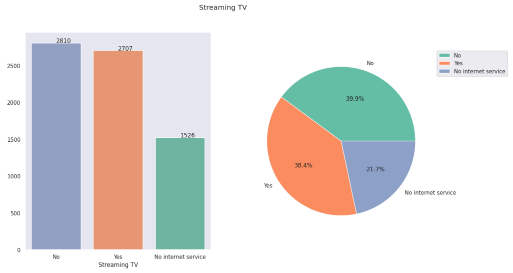
```
tech_support_counts = df['TechSupport'].value_counts()
x = tech_support_counts.index
y = tech_support_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(tech_support_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('TechSupport')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('TechSupport')
plt.show()
```

```
<ipython-input-69-3ac281d69399>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0
```



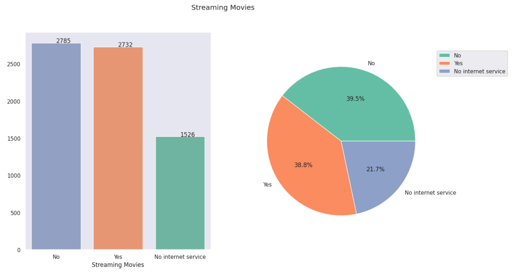
```
streaming_tv_counts = df['StreamingTV'].value_counts()
x = streaming_tv_counts.index
y = streaming_tv_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(streaming_tv_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('Streaming TV')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Streaming TV')
plt.show()
```

```
<ipython-input-72-7f09920d29f0>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0
```



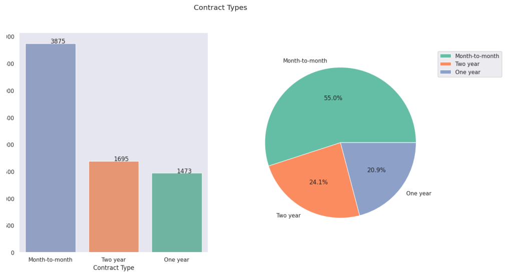
```
streaming_movies_counts = df['StreamingMovies'].value_counts()
x = streaming_movies_counts.index
y = streaming_movies_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(streaming_movies_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('Streaming Movies')

# Pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Streaming Movies')
plt.show()
```

```
<ipython-input-76-d6af87b8fa48>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[6
```



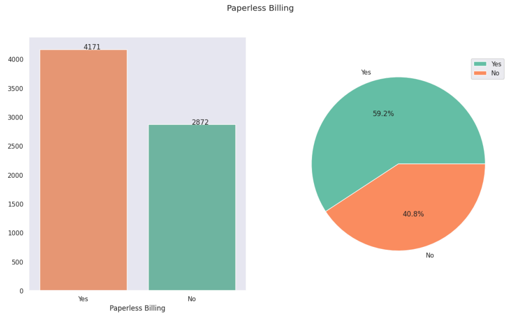
```
contract_counts = df['Contract'].value_counts()
x = contract_counts.index
y = contract_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(contract_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('Contract Type')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Contract Types')
plt.show()
```

```
 ipython-input-73-78dc6306a91d>:10: FutureWarning:

 assing `palette` without assigning `hue` is deprec

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0]
```



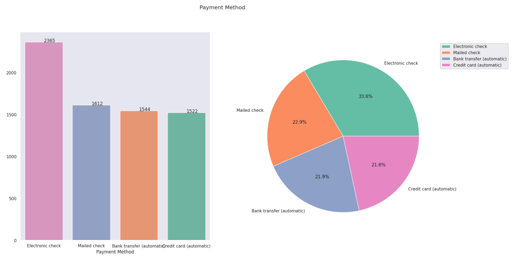
```
paperless_billing_counts = df['PaperlessBilling'].value_counts()
x = paperless_billing_counts.index
y = paperless_billing_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(paperless_billing_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('Paperless Billing')

# Pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Paperless Billing')
plt.show()
```

```
<ipython-input-75-0ee52f832f03>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0
```



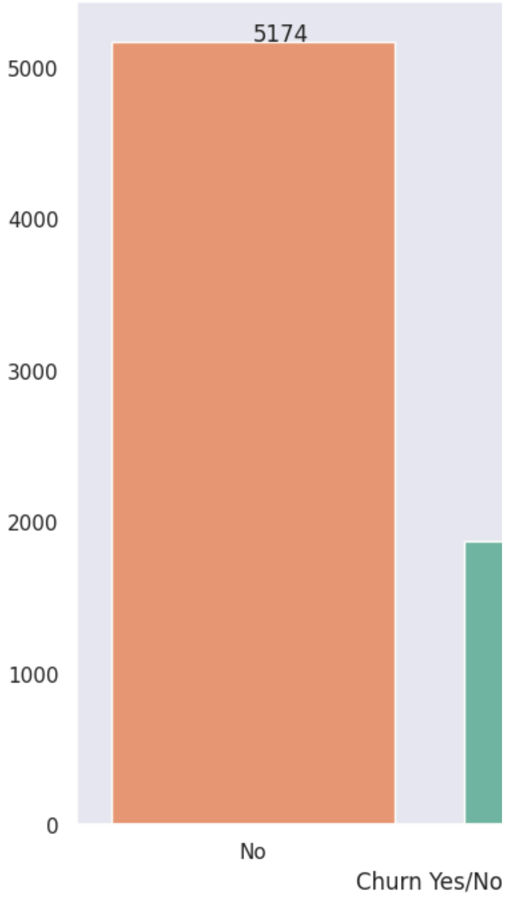
```
payment_method_counts = df['PaymentMethod'].value_counts()
x = payment_method_counts.index
y = payment_method_counts.values

fig, ax = plt.subplots(1, 2, figsize=(20, 10))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(payment_method_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('Payment Method')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Payment Method')
plt.show()
```

```
<ipython-input-79-67ecd2e2834b>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[
```



```
churn_counts = df['Churn'].value_counts()
x = churn_counts.index
y = churn_counts.values

fig, ax = plt.subplots(1, 2, figsize=(15, 8))

#bar plot
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("Set2", len(churn_counts))
sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[0])

for p in ax[0].patches:
    ax[0].annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.get_

ax[0].set_xlabel('Churn Yes/No')

#pie chart
ax[1].pie(y, labels=x, colors=pal, autopct='%1.1f%%')

#legend and title
plt.legend(bbox_to_anchor=(1, 1))
plt.suptitle('Churn Yes/No')
plt.show()
```

```
<ipython-input-80-fdaa9f2431d5>:10: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x=x, y=y, palette=pal[::-1], ax=ax[
```



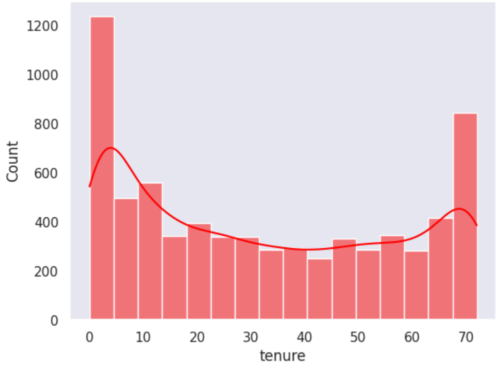
NUMERICAL COLUMNS EXPLORATION

```
sns.histplot(x = df['tenure'],kde = True, sns.color_palette='Set2')
import seaborn as sns
import matplotlib.pyplot as plt

sns.histplot(x = df['tenure'], kde = True, palette='Set2')
plt.show()
```

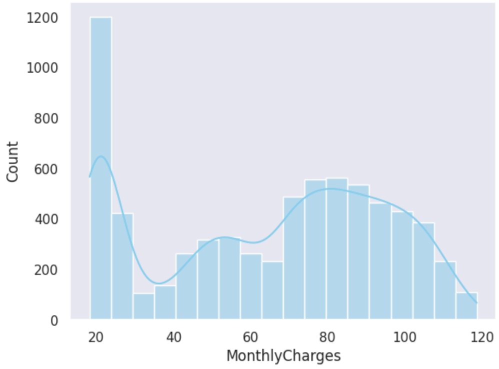
✓ ✕

```
---------------------------------------------------
-------------------------
SyntaxError
Traceback (most recent call last)
/usr/local/lib/python3.10/dist-
packages/IPython/core/compilerop.py in
ast_parse(self, source, filename, symbol)
     99         Arguments are exactly the same as
ast.parse (in the standard library),
    100         and are passed to the built-in
compile function."""
--> 101         return compile(source, filename,
symbol, self.flags | PyCF_ONLY_AST, 1)
    102
    103     def reset_compiler_flags(self):
```

```
sns.histplot(x=df['tenure'], kde=True, color='red')
plt.show()
```



```
sns.histplot(x = df['MonthlyCharges'],kde = True, color='skyblue')
plt.show()
```



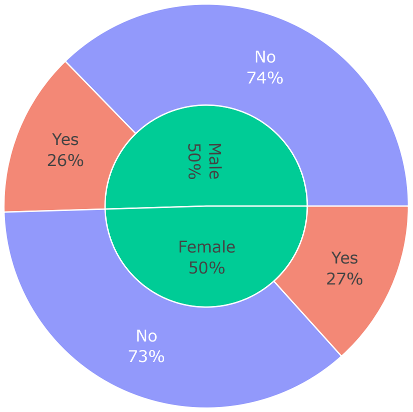
```
print(df.columns)
```

```
Index(['Partner', 'Churn'], dtype='object')
```

Start coding or generate with AI.

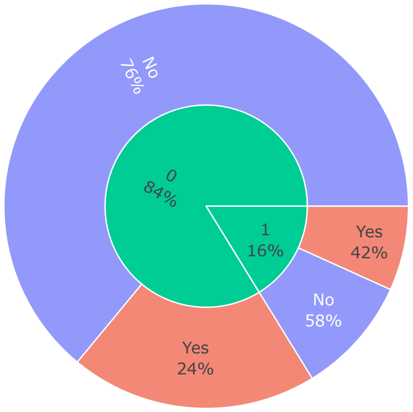
```
fig = px.sunburst(data_frame=df,
                  path=['gender', 'Churn'],
                  color='Churn',
                  title='Gender vs Churn'
                 )

fig.update_traces(textinfo='label+percent parent')
fig.update_layout(margin=dict(t=40, l=0, r=0, b=0))
fig.show()
```

Gender vs Churn



```
fig = px.sunburst(data_frame=df,
                  path=['SeniorCitizen', 'Churn'],
                  color='Churn',
                  title='SeniorCitizen vs Churn'
                 )

fig.update_traces(textinfo='label+percent parent')
fig.update_layout(margin=dict(t=40, l=0, r=0, b=0))
fig.show()
```

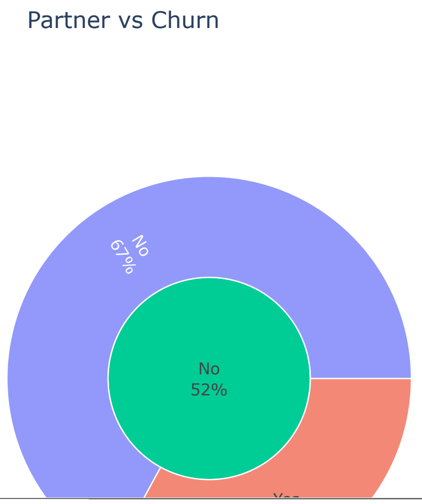
SeniorCitizen vs Churn



Generate 10 random numbers using numpy Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

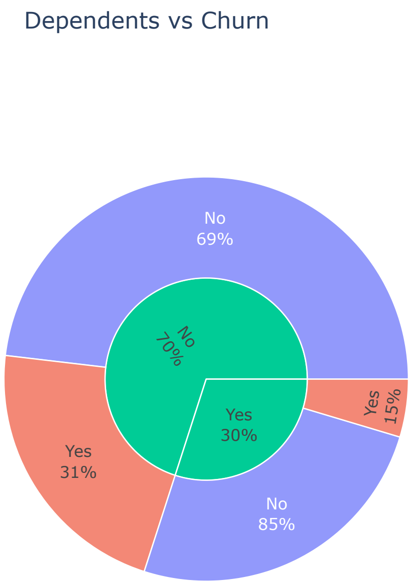
```
fig = px.sunburst(data_frame=df,
                  path=['Partner', 'Churn'],
                  color='Churn',
                  title='Partner vs Churn'
                 )

fig.update_traces(textinfo='label+percent parent')
fig.update_layout(margin=dict(t=40, l=0, r=0, b=0))
fig.show()
```



Generate randomly select 5 items from a list Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

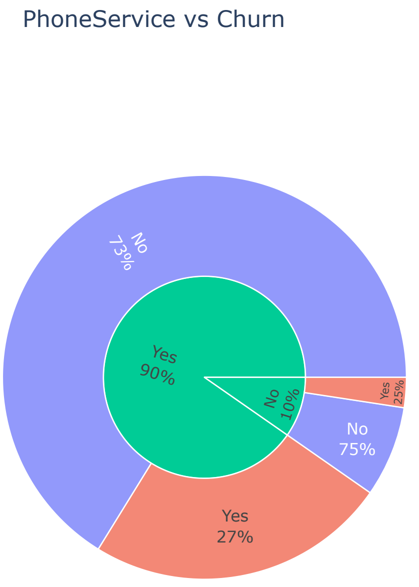
```
fig = px.sunburst(data_frame=df,
                  path=['Dependents', 'Churn'],
                  color='Churn',
                  title='Dependents vs Churn'
                 )

fig.update_traces(textinfo='label+percent parent')
fig.update_layout(margin=dict(t=40, l=0, r=0, b=0))
fig.show()
```



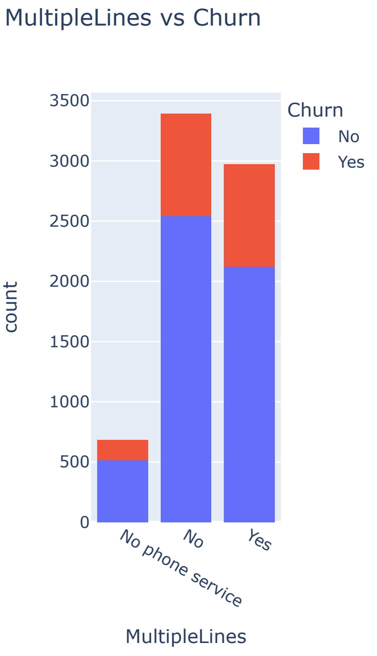
Generate print hello world using rot13 Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

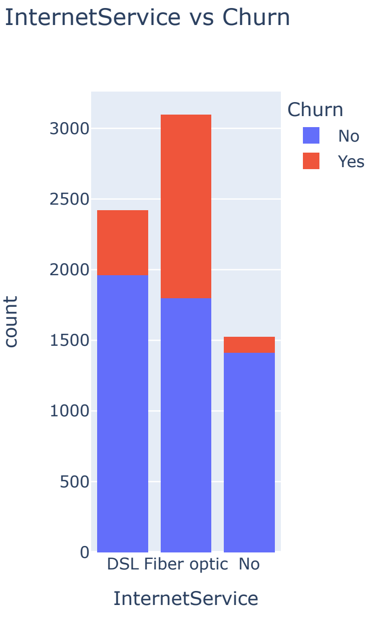
```
fig = px.sunburst(data_frame=df,
                  path=['PhoneService', 'Churn'],
                  color='Churn',
                  title='PhoneService vs Churn'
                 )

fig.update_traces(textinfo='label+percent parent')
fig.update_layout(margin=dict(t=40, l=0, r=0, b=0))
fig.show()
```



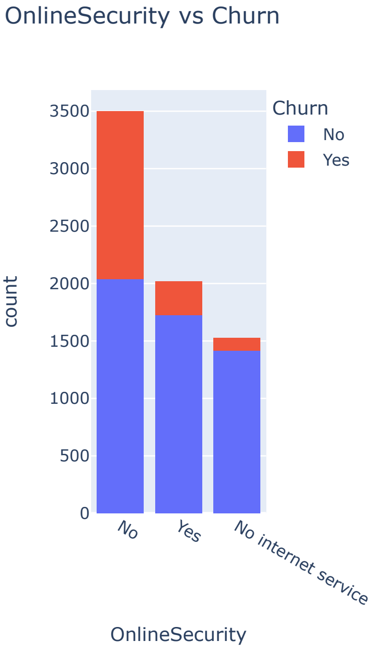
```
fig = px.histogram(data_frame = df,
             x = "MultipleLines",
             color="Churn", title="MultipleLines vs Churn")

fig.show()
```



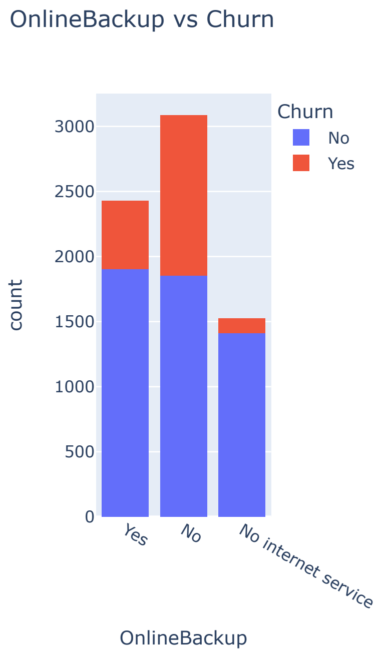
```
fig = px.histogram(data_frame = df,
             x = "InternetService",
             color="Churn", title="InternetService vs Churn")

fig.show()
```



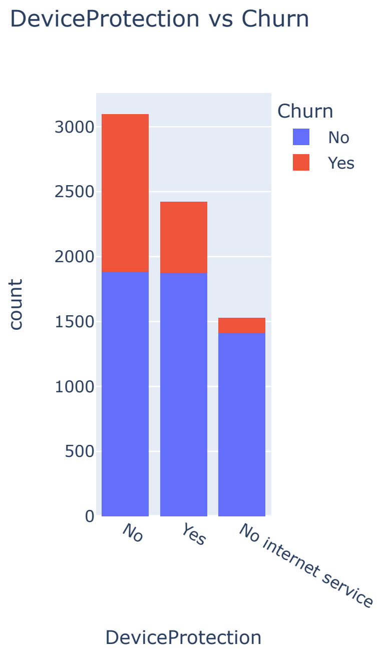
```
fig = px.histogram(data_frame = df,
             x = "OnlineSecurity",
             color="Churn", title="OnlineSecurity vs Churn")

fig.show()
```



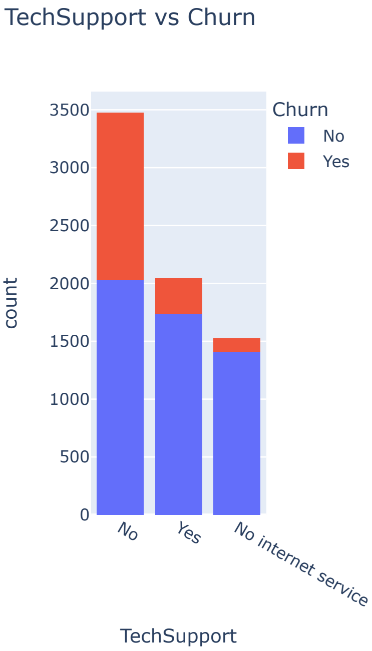
```
fig = px.histogram(data_frame = df,
             x = "OnlineBackup",
             color="Churn", title="OnlineBackup vs Churn")

fig.show()
```



```
fig = px.histogram(data_frame = df,
             x = "DeviceProtection",
             color="Churn", title="DeviceProtection vs Churn")

fig.show()
```



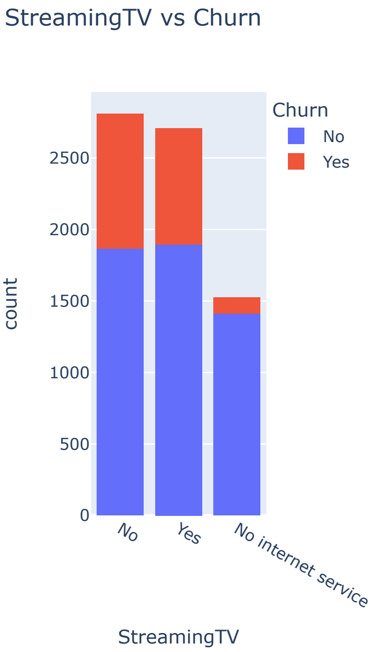
```
fig = px.histogram(data_frame = df,
             x = "TechSupport",
             color="Churn", title="TechSupport vs Churn")

fig.show()
```


TechSupport vs Churn

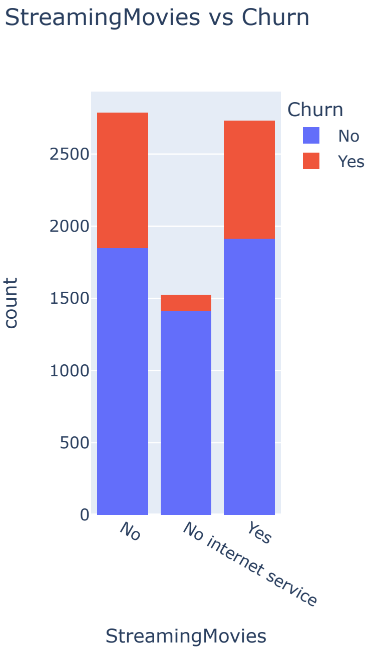
Generate randomly select 5 items from a list Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

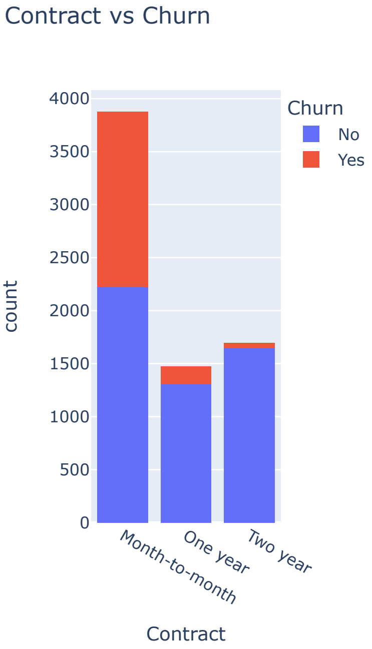
```
fig = px.histogram(data_frame = df,
             x = "StreamingTV",
             color="Churn", title="StreamingTV vs Churn")

fig.show()
```


StreamingTV vs Churn

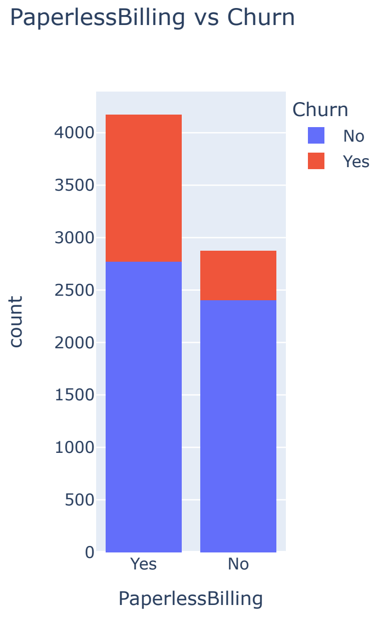
```
fig = px.histogram(data_frame = df,
             x = "StreamingMovies",
             color="Churn", title="StreamingMovies vs Churn")

fig.show()
```



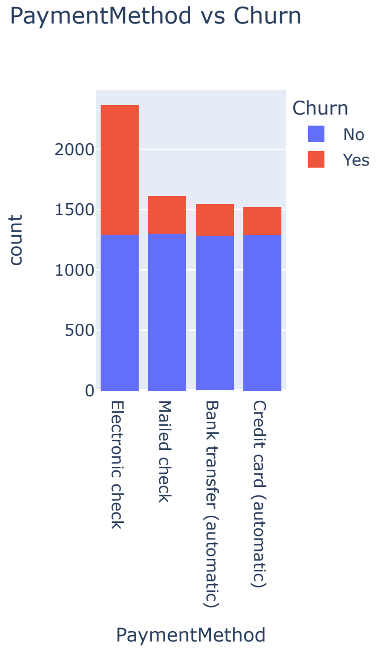
```
fig = px.histogram(data_frame = df,
             x = "Contract",
             color="Churn", title="Contract vs Churn")

fig.show()
```



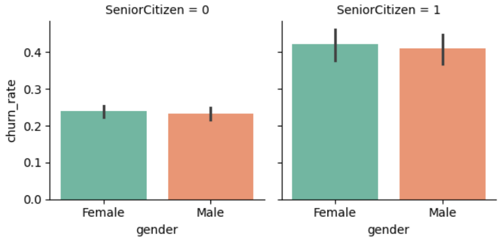
```
fig = px.histogram(data_frame = df,
             x = "PaperlessBilling",
             color="Churn", title="PaperlessBilling vs Churn")

fig.show()
```



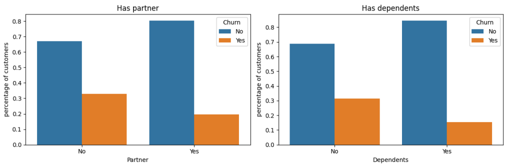
```
fig = px.histogram(data_frame = df,
             x = "PaymentMethod",
             color="Churn", title="PaymentMethod vs Churn")

fig.show()
```



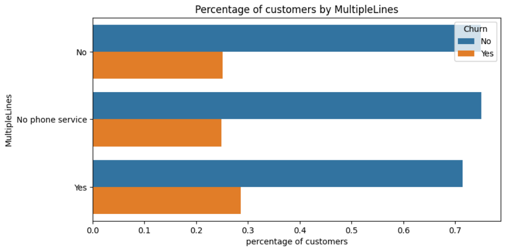
```
df['churn_rate'] = df['Churn'].replace("No", 0).replace("Yes", 1)
g = sns.FacetGrid(df, col="SeniorCitizen")
ax = g.map(sns.barplot, "gender", "churn_rate", palette = "Set2", order= ['Female',
     /usr/local/lib/python3.10/dist-packages/seaborn/ax

     Passing `palette` without assigning `hue` is depre

       func(*plot_args, **plot_kwargs)
     /usr/local/lib/python3.10/dist-packages/seaborn/ax

     Passing `palette` without assigning `hue` is depre

       func(*plot_args, **plot_kwargs)
```



```
fig, axis = plt.subplots(1, 2, figsize=(12, 4))

axis_titles = ["Has partner", "Has dependents"]
columns = ['Partner', 'Dependents']
axis_y = "percentage of customers"

for ax, col, title in zip(axis, columns, axis_titles):

    gp = df.groupby(col)["Churn"].value_counts(normalize=True).rename(axis_y).reset

    #plotting
    sns.barplot(x=col, y=axis_y, hue='Churn', data=gp, ax=ax)
    ax.set_title(title)

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



```
def barplot_percentages(column, orient='v'):

    gp = df.groupby(column)["Churn"].value_counts(normalize=True).rename("percentag

    if orient == 'h':
        sns.barplot(y=column, x="percentage of customers", hue='Churn', data=gp, or
    else:
        sns.barplot(x=column, y="percentage of customers", hue='Churn', data=gp, or

    plt.title(f"Percentage of customers by {column}")

plt.figure(figsize=(9, 4.5))
barplot_percentages("MultipleLines", orient='h')
plt.show()
```



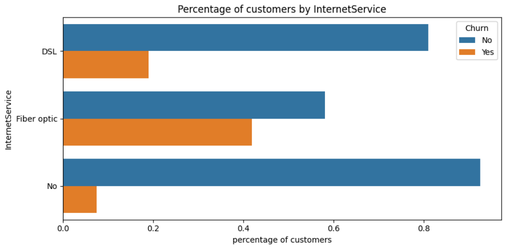
Generate a slider using jupyter widgets Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

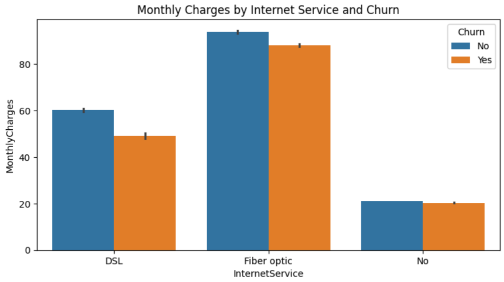
```
def barplot_percentages(column, orient='v'):

    gp = df.groupby(column)["Churn"].value_counts(normalize=True).rename("percentag

    if orient == 'h':
        sns.barplot(y=column, x="percentage of customers", hue='Churn', data=gp, or
    else:
        sns.barplot(x=column, y="percentage of customers", hue='Churn', data=gp, or

    plt.title(f"Percentage of customers by {column}")

plt.figure(figsize=(9, 4.5))
barplot_percentages("MultipleLines", orient='h')
plt.show()
```

```
<Figure size 900x450 with 0 Axes>
```

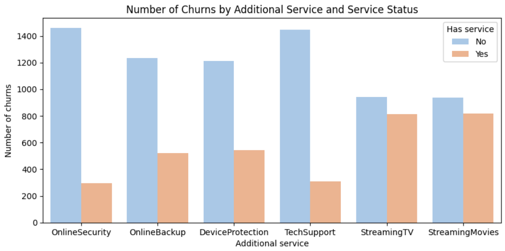
```
def barplot_percentages(column, orient='v'):

    gp = df.groupby(column)["Churn"].value_counts(normalize=True).rename("percentag

    if orient == 'h':
        sns.barplot(y=column, x="percentage of customers", hue='Churn', data=gp, or
    else:
        sns.barplot(x=column, y="percentage of customers", hue='Churn', data=gp, or

    plt.title(f"Percentage of customers by {column}")
    plt.tight_layout()

plt.figure(figsize=(9, 4.5))
barplot_percentages("InternetService", orient="h")
plt.show()
```



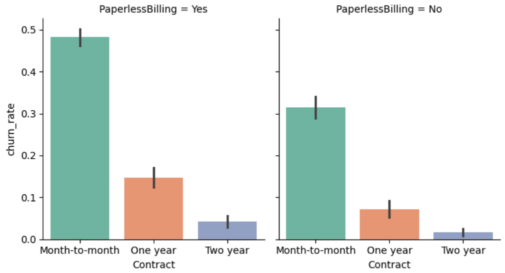
```
plt.figure(figsize=(9, 4.5))
ax = sns.barplot(x="InternetService", y="MonthlyCharges", hue="Churn", data=df)

plt.title("Monthly Charges by Internet Service and Churn")

plt.show()
```



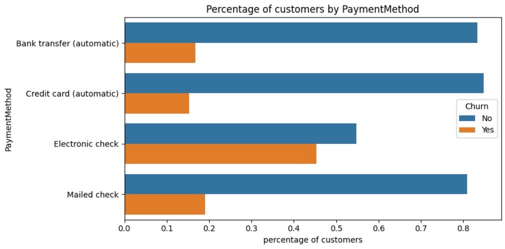
```
cols = ["OnlineSecurity", "OnlineBackup", "DeviceProtection", "TechSupport", "Strea
df1 = df[(df.InternetService != "No") & (df.Churn == "Yes")]
df1 = pd.melt(df1[cols]).rename({'value': 'Has service'}, axis=1)
plt.figure(figsize=(10, 4.5))
ax = sns.countplot(data=df1, x='variable', hue='Has service', hue_order=['No', 'Yes

ax.set(xlabel='Additional service', ylabel='Number of churns')
ax.set_title('Number of Churns by Additional Service and Service Status')

plt.show()
```



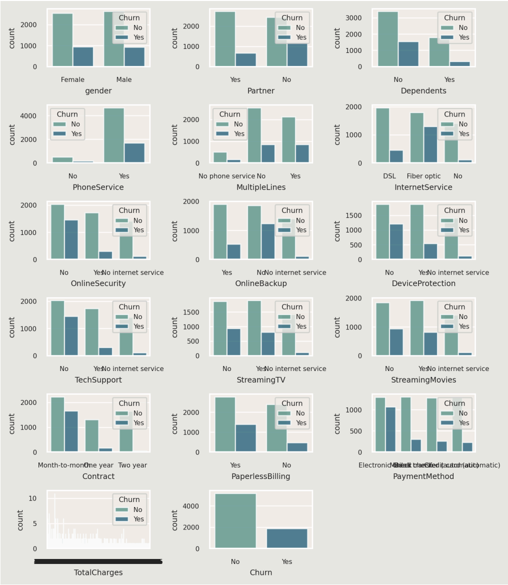
```
g = sns.FacetGrid(df, col="PaperlessBilling", height=4, aspect=.9)
ax = g.map(sns.barplot, "Contract", "churn_rate", palette = "Set2", order= ['Month-
     /usr/local/lib/python3.10/dist-packages/seaborn/a

     Passing `palette` without assigning `hue` is depre

       func(*plot_args, **plot_kwargs)
     /usr/local/lib/python3.10/dist-packages/seaborn/a

     Passing `palette` without assigning `hue` is depre

       func(*plot_args, **plot_kwargs)
```



```
plt.figure(figsize=(9, 4.5))
barplot_percentages("PaymentMethod", orient='h',)
```



```
train_cat_visual_1 = df.select_dtypes(
                     include = ['object', 'category']).columns.tolist()
train_cat_visual_1.remove('customerID')


sns.set_theme(rc = {'figure.dpi': 250, 'axes.labelsize': 7,
                    'axes.facecolor': '#f0eee9', 'grid.color': '#fffdfa',
                    'figure.facecolor': '#e8e6e1'}, font_scale = 0.55)

fig, ax = plt.subplots(6, 3, figsize = (6.5, 7.5))

for indx, (column, axes) in list(enumerate(list(zip(train_cat_visual_1,
                                                    ax.flatten())))):

    sns.countplot(ax = axes, x = df[column], hue = df['Churn'],
                  palette = 'crest', alpha = 0.8)

else:
    [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]

axes_legend = ax.flatten()

axes_legend[1].legend(title = 'Churn', loc = 'upper right')
axes_legend[2].legend(title = 'Churn', loc = 'upper right')

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



```
train_num_visual_0 = ['MonthlyCharges', 'tenure','TotalCharges']
```

OUTLIER ANALYSIS

```
columns_to_check = ['tenure', 'MonthlyCharges']

def count_outliers(data, col):
    q1 = data[col].quantile(0.25)
    q3 = data[col].quantile(0.75)
    iqr = q3 - q1
    lower_limit = q1 - 1.5 * iqr
    upper_limit = q3 + 1.5 * iqr

    outliers_below = data[data[col] < lower_limit][col].size
    outliers_above = data[data[col] > upper_limit][col].size
    total_outliers = outliers_below + outliers_above

    if total_outliers == 0:
        print(f"No outliers in {col}")
    else:
        print(f"There are outliers in {col}")
        print(f'Count of outliers: {total_outliers}')

for col in columns_to_check:
    count_outliers(df, col)
```

```
No outliers in tenure
No outliers in MonthlyCharges
```

```
df.drop(['customerID','Churn'],axis = 1,inplace = True)
```

```
df
```

```
      gender  SeniorCitizen  Partner  Dependents
0     Female              0      Yes          No
1       Male              0       No          No
2       Male              0       No          No
3       Male              0       No          No
4     Female              0       No          No
...      ...            ...      ...         ...
7038    Male              0      Yes         Yes
7039  Female              0      Yes         Yes
```

Gemini access in Colab is available to unsubscribed users for a limited time only. Upgrade to Colab Pro

Enter a prompt here

0 / 400

Responses may display inaccurate or offensive information that doesn't represent Google's views. Learn more