SAIKET SYSTEMS INTERNSHIP SOLUTION

TOOL USED: PYTHON

NAME: ADIGUN SAMUEL

ADIGUN SAMUEL

SAIKET SYSTEM (BUSINESS ANALYSIS)

Contents

TASK1: Understand the Dataset Description:

- ❖ Familiarize yourself with the dataset.
  Steps:
- ❖ Load the dataset using pandas.
- ❖ Display the first 10 rows.
- ❖ Identify the data types of each column.
- ❖ Check for missing values.

SOLUTION

✓ Loading the dataset using pandas

```
[15] import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns

     # Option to display all rows
     pd.set_option('display.max_rows', None)

     # Loading the Excel file
     df = pd.read_excel('Telco_Customer_Churn_Dataset.xlsx')

     # data cleaning and preparation
     df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
     df['TotalCharges'] = df['TotalCharges'].fillna(0)
     df.columns = df.columns.str.lower().str.replace(' ', '_')
     df['churn'] = df['churn'].map({'Yes': 1, 'No': 0})
```

✓ Display the first 10 rows.

```
[16] #  To Show first 10 rows
     display(df.head(10))
```

	customerid	gender	seniorcitizen	partner	dependents	tenure	phoneservice	multiplelines	internetservice	onlinesecurity	...	deviceprotection	techsupport	streamingtv	streamingmovies	contract	paperlessbilling	paymentmethod
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month-to-month	Yes	Electronic check
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year	No	Mailed check
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month-to-month	Yes	Mailed check
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year	No	Bank transfer (automatic)
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month-to-month	Yes	Electronic check
5	9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	...	Yes	No	Yes	Yes	Month-to-month	Yes	Electronic check
6	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	...	No	No	Yes	No	Month-to-month	Yes	Credit card (automatic)
7	6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL	Yes	...	No	No	No	No	Month-to-month	No	Mailed check
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	...	Yes	Yes	Yes	Yes	Month-to-month	Yes	Electronic check
9	6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	...	No	No	No	No	One year	No	Bank transfer (automatic)

10 rows × 21 columns

- **Identification of the data types of each column**

```
print(df.dtypes)
```

```
customerID          object
gender              object
SeniorCitizen        int64
Partner             object
Dependents          object
tenure               int64
PhoneService        object
MultipleLines       object
InternetService     object
OnlineSecurity      object
OnlineBackup        object
DeviceProtection    object
TechSupport         object
StreamingTV         object
StreamingMovies     object
Contract            object
PaperlessBilling    object
PaymentMethod       object
MonthlyCharges     float64
TotalCharges       float64
Churn               object
dtype: object
```

- Customer ID – **object**
- Gender – **object**
- Senior Citizen – **int64**
- Partner – **object**
- Dependents – **object**
- Tenure – **int64**
- Phone Service – **object**
- Multiple Lines – **object**
- Internet Service – **object**
- Online Security – **object**
- Online Backup – **object**
- Device Protection – **object**
- Tech Support – **object**
- Streaming TV – **object**
- Streaming Movies – **object**
- Contract – **object**
- Paperless Billing – **object**
- Payment Method – **object**
- Monthly Charges – **float64**
- Total Charges – **float64**
- Churn – **object**

❖ **Check for missing values.**

```
print(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       11
Churn               0
dtype: int64
```

- **No Missing Data in Most Columns** – All columns except Total Charges have 0 missing values, meaning they are complete and ready for analysis.
- **Total Charges Has Missing Values** – This column has **11 missing entries**.

TASK 2: DATA CLEANING

QUESTION:

- ❖ Handle missing values appropriately (e.g., fill, drop, or impute).
- ❖ Remove duplicate records if any.
- ❖ Standardize column names (convert to lowercase and replace spaces with underscores)

ANALYSIS & SOLUTION

```
[ ] df['TotalCharges'] = df['TotalCharges'].fillna(0)
```

```
df[df['TotalCharges'] == 0]
```

Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	Yes	Yes	No	Two year	Yes	Bank transfer (automatic)	52.55	0.0	No
No	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	No internet service	No internet service	Two year	No	Mailed check	20.25	0.0	No
Yes	Yes	0	Yes	No	DSL	Yes	...	Yes	No	Yes	Yes	Two year	No	Mailed check	80.85	0.0	No
Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No internet service	No internet service	No internet service	Two year	No	Mailed check	25.75	0.0	No
Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	Yes	Yes	No	Two year	No	Credit card (automatic)	56.05	0.0	No
Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	No internet service	No internet service	Two year	No	Mailed check	19.85	0.0	No
Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No internet service	No internet service	No internet service	Two year	No	Mailed check	25.35	0.0	No
Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	No internet service	No internet service	Two year	No	Mailed check	20.00	0.0	No
Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	No internet service	No internet service	One year	Yes	Mailed check	19.70	0.0	No
Yes	Yes	0	Yes	Yes	DSL	No	...	Yes	Yes	Yes	No	Two year	No	Mailed check	73.35	0.0	No
No	Yes	0	Yes	Yes	DSL	Yes	...	No	Yes	No	No	Two year	Yes	Bank transfer (automatic)	61.90	0.0	No

✓ **Handling missing values appropriately**

Note: I replaced the 11 missing values in the **`Total Charges`** column with zero using a python function to do it. This ensures the dataset is complete and helps maintain accuracy when performing deeper analyses, such as the ones stated later in the task

✓ **Standardize column names (convert to lowercase and replace spaces with underscores)**

```
[ ] df.columns = df.columns.str.lower().str.replace(' ', '_')
```

```
print(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')
```

Note: The column names were convert to lowercase using a Python function, as shown in the uploaded image, to ensure consistency when running commands.

TASK 3: EXPLORATORY DATA ANALYSIS (EDA)

- Understand trends and distributions in the data.
- Generate summary statistics (mean, median, and mode).
- Create visualizations for numerical columns (histograms, box plots).
- Analyze churn rates (e.g., churn vs. non-churn proportions)

ANALYSIS & SOLUTION

✓ **Generate summary statistics (mean, median, and mode).**

```
[ ] df.describe()
```

	seniorcitizen	tenure	monthlycharges	totalcharges
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692	2279.734304
std	0.368612	24.559481	30.090047	2266.794470
min	0.000000	0.000000	18.250000	0.000000
25%	0.000000	9.000000	35.500000	398.550000
50%	0.000000	29.000000	70.350000	1394.550000
75%	0.000000	55.000000	89.850000	3786.600000

{} Variables   Terminal

```
[ ] df.describe()
```

	seniorcitizen	tenure	monthlycharges	totalcharges
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692	2279.734304
std	0.368612	24.559481	30.090047	2266.794470
min	0.000000	0.000000	18.250000	0.000000
25%	0.000000	9.000000	35.500000	398.550000
50%	0.000000	29.000000	70.350000	1394.550000
75%	0.000000	55.000000	89.850000	3786.600000
max	1.000000	72.000000	118.750000	8684.800000

```
[ ] df.mean(numeric_only=True)
```

```
df.median(numeric_only=True)
```

	0
seniorcitizen	0.00
tenure	29.00
monthlycharges	70.35
totalcharges	1394.55

dtype: float64

```
df.mean(numeric_only=True)
```

	0
seniorcitizen	0.162147
tenure	32.371149
monthlycharges	64.761692
totalcharges	2279.734304

dtype: float64

Summary of Tenure, Senior Citizen, Monthly Charges, and Total Charges

1. **Senior Citizen**
    - **Mean**: 0.16 → about 16% of the customers are senior citizens.
    - Since the median and 75% values are **0**, this confirms that the majority of customers are *not* senior citizens.

2. **Tenure**
    - **Mean**: 32.37 months → the average customer stays for about 2 years and 8 months.
    - **Median**: 29 months → half of the customers have stayed less than 29 months.
    - **Max**: 72 months → some customers have been with the company for the full 6 years.

3. **Monthly Charges**
    - **Mean**: $64.76 → Average monthly bill is around $65.
    - **Median**: $70.35 → half of customers pay less than this amount.
    - The spread (min $18.25 to max $118.75) shows a wide range in plans and services.

4. **Total Charges**
    - **Mean**: $2279.73 → on average, a customer has paid about $2,280 in total.
    - **Median**: $1,394.55 → half of the customers have paid less than this over their tenure.
    - The large range (0 to $8,684.80) suggests big differences in how long customers stay and what plans they choose.

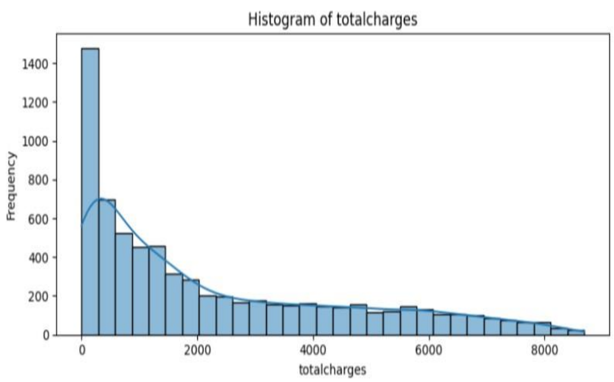
Key Observations

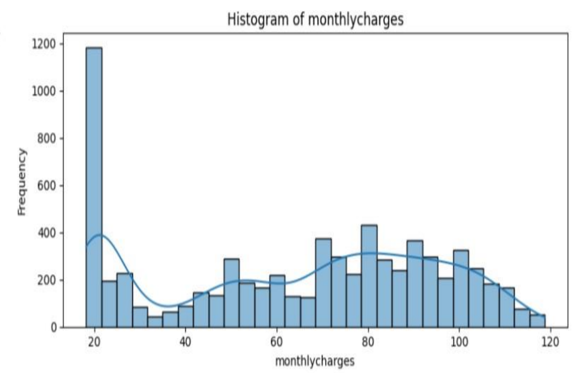
- The majority of customers are **not** senior citizens, but those who are may have different usage or churn behavior.
- Customers staying longer (higher tenure) naturally have higher total charges.
- Monthly charges vary greatly, likely due to differences in service packages, contract lengths, and optional add-ons.
- A small proportion of customers have zero total charges, possibly due to new sign-ups or incomplete billing data.

- **Create visualizations for numerical columns (histograms, box plots)**.

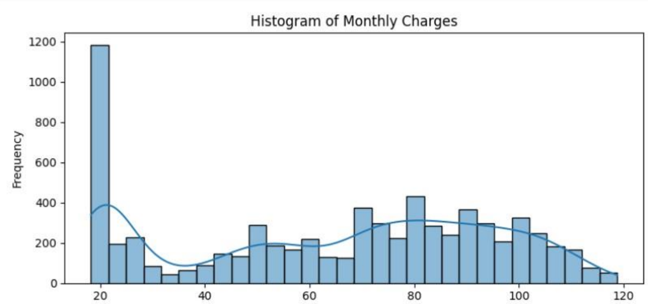
```
[ ] import matplotlib.pyplot as plt
    import seaborn as sns
```

```
[ ] numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
    print(numerical_cols)
```

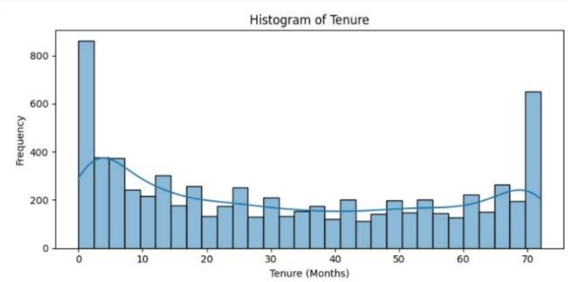




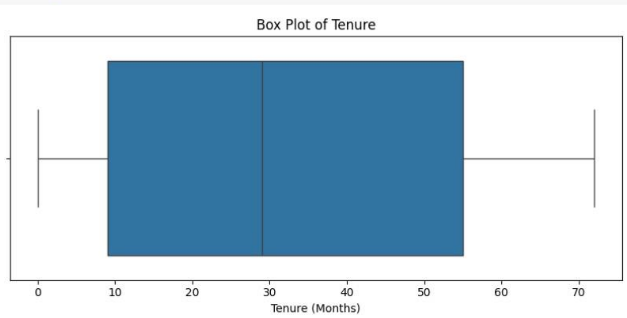

```
plt.figure(figsize=(8, 4))
sns.histplot(df['monthlycharges'], bins=30, kde=True)
plt.title('Histogram of Monthly Charges')
plt.xlabel('Monthly Charges')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



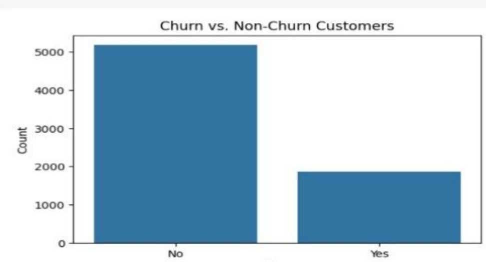

```
[ ] plt.figure(figsize=(8, 4))
    sns.histplot(df['tenure'], bins=30, kde=True)
    plt.title('Histogram of Tenure')
    plt.xlabel('Tenure (Months)')
    plt.ylabel('Frequency')
    plt.tight_layout()
    plt.show()
```



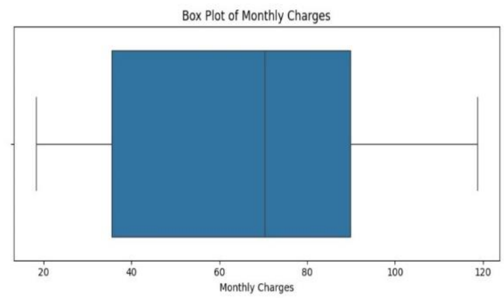

```
plt.figure(figsize=(8, 4))
sns.boxplot(x=df['tenure'])
plt.title('Box Plot of Tenure')
plt.xlabel('Tenure (Months)')
plt.tight_layout()
plt.show()
```



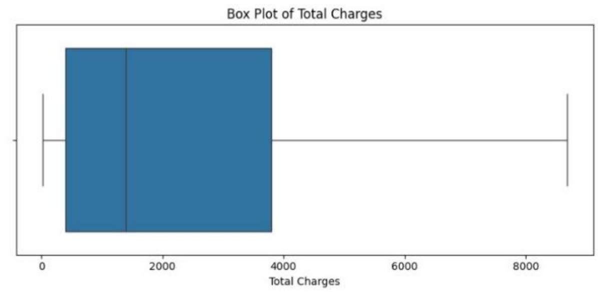

```
plt.figure(figsize=(6,4))
sns.countplot(x='churn', data=df)
plt.title('Churn vs. Non-Churn Customers')
plt.xticks([0, 1], ['No', 'Yes'])
plt.xlabel('Churn')
plt.ylabel('Count')
plt.show()
```

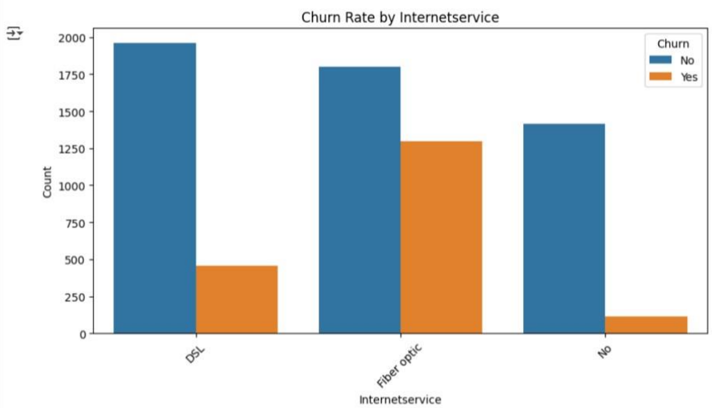


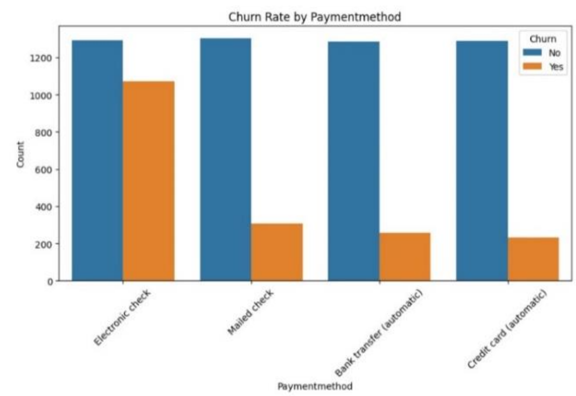
```python
plt.figure(figsize=(8, 4))
sns.boxplot(x=df['monthlycharges'])
plt.title('Box Plot of Monthly Charges')
plt.xlabel('Monthly Charges')
plt.tight_layout()
plt.show()
```



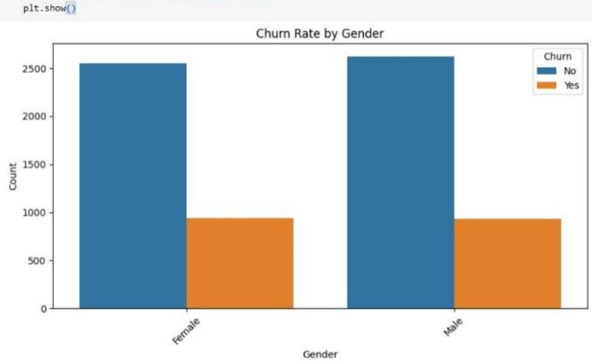

```python
plt.figure(figsize=(8, 4))
sns.boxplot(x=df['totalcharges'])
plt.title('Box Plot of Total Charges')
plt.xlabel('Total Charges')
plt.tight_layout()
plt.show()
```

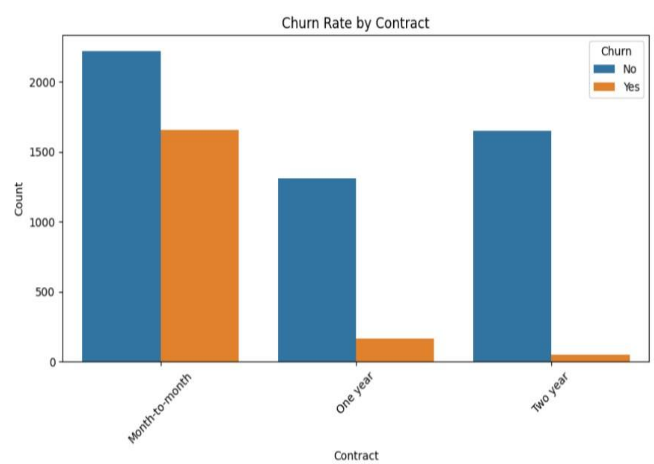







```python
for col in ['gender', 'internetservice', 'contract', 'paymentmethod']:
    plt.figure(figsize=(10, 5))
    sns.countplot(x=col, hue='churn', data=df)
    plt.title(f'Churn Rate by {col.title()}')
    plt.xlabel(col.title())
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.legend(title='Churn', labels=['No', 'Yes'])
    plt.show()
```



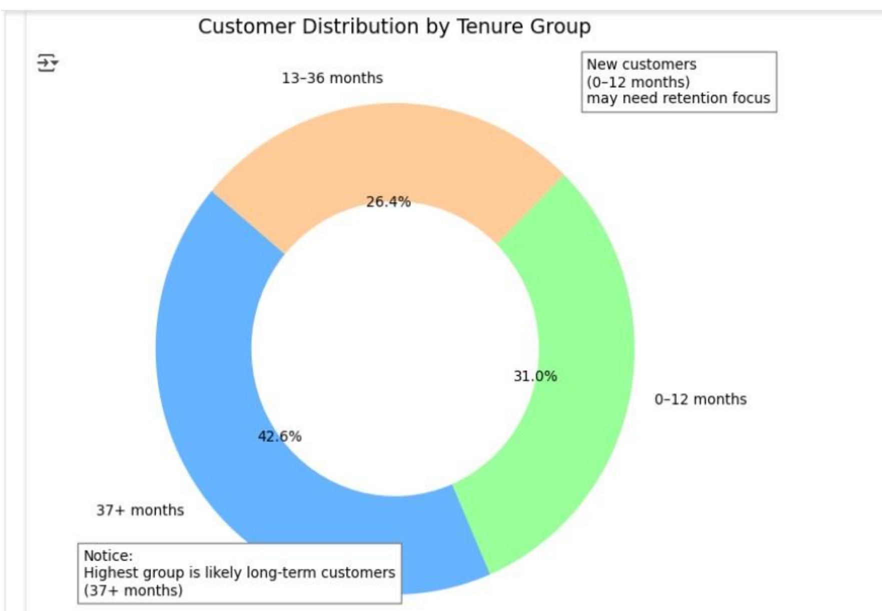


TASK 4: CUSTOMER SEGMENTATION VISUALIZATION

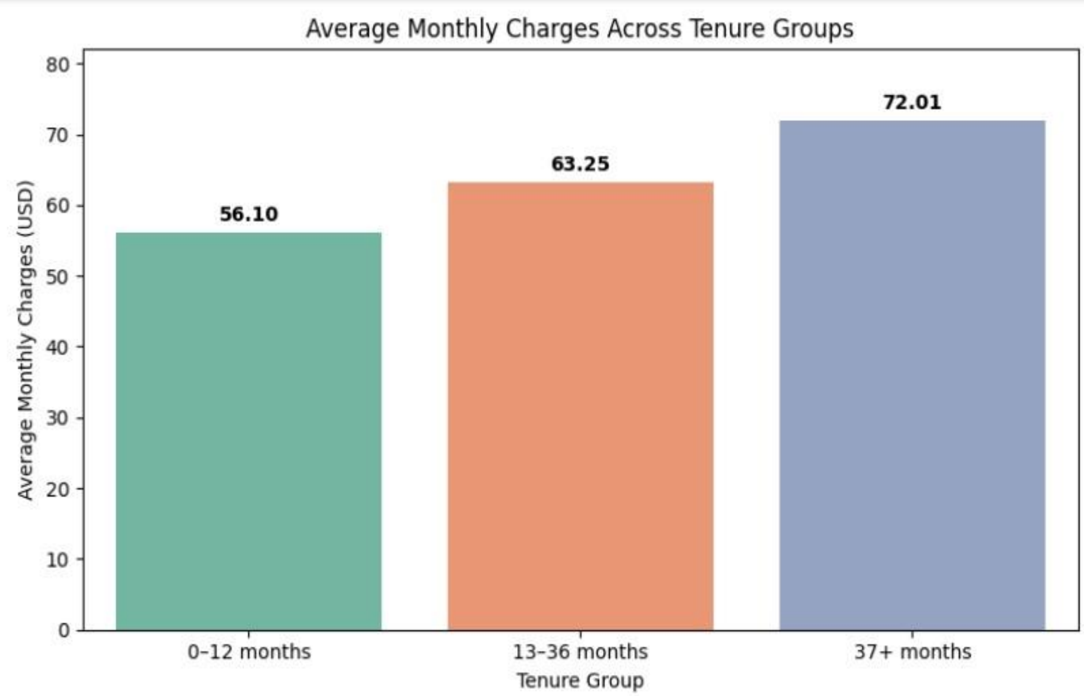
- Visualize customer distribution by tenure using pie or donut charts. (e.g., 0-12 months, 13-36 months, 37+ months).
- Use a clustered bar chart to compare average monthly charges across tenure categories,
- adding annotations to highlight significant trends

```
[ ] tenure_counts = df['tenure_group'].value_counts()
```

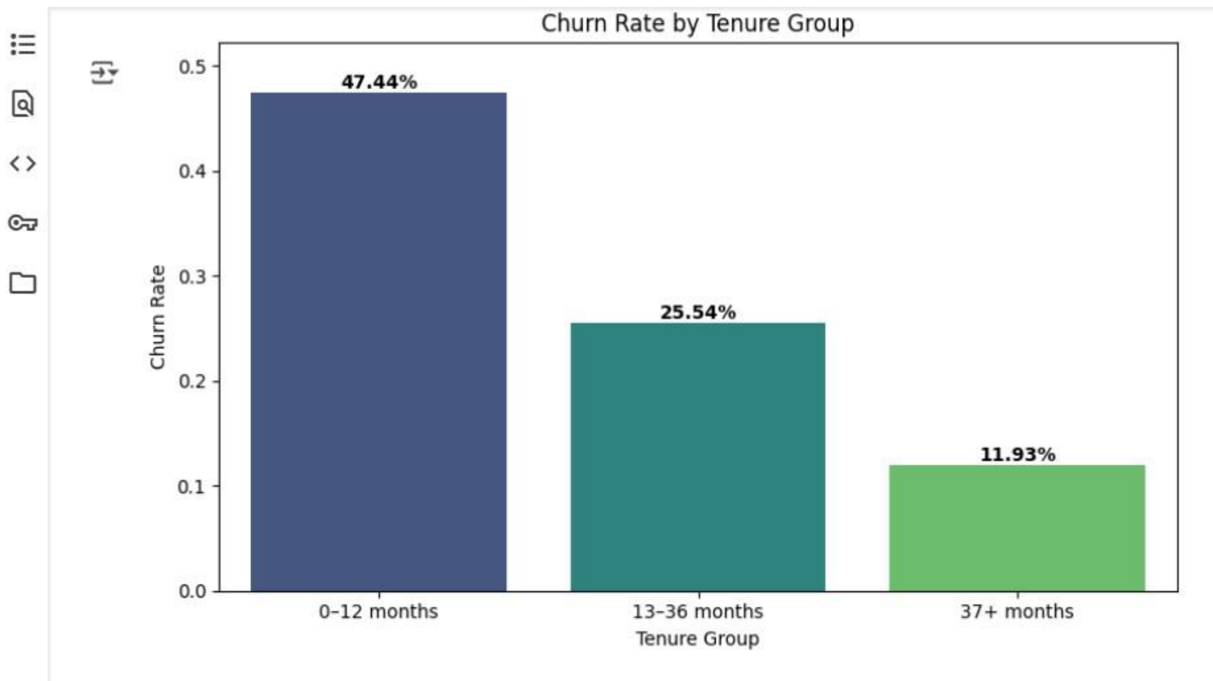
```
import matplotlib.pyplot as plt

tenure_counts = df['tenure_group'].value_counts()
colors = ['#66b3ff', '#99ff99', '#ffcc99']
plt.figure(figsize=(7, 7))
wedges, texts, autotexts = plt.pie(
    tenure_counts,
    labels=tenure_counts.index,
    autopct='%1.1f%%',
    startangle=140,
    colors=colors,
    wedgeprops=dict(width=0.4)
)
plt.title('Customer Distribution by Tenure Group', fontsize=14)
plt.text(-1.3, -1.0, 'Notice:\nHighest group is likely long-term customers\n(37+ months)', fontsize=10, bbox=dict(facecolor='white', edgecolor='gray'))
plt.text(0.8, 1.0, 'New customers\n(0-12 months)\nmay need retention focus', fontsize=10, bbox=dict(facecolor='white', edgecolor='gray'))
plt.tight_layout()
plt.show()
```



```python
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 5))
bars = sns.barplot(x='Tenure Group', y='Churn_Rate', data=group_stats, hue='Tenure Group', palette='viridis', legend=False)

# Add value annotations on each bar
for bar in bars.patches:
    height = bar.get_height()
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        height,
        f'{height:.2%}',
        ha='center',
        va='bottom',
        fontsize=10,
        fontweight='bold'
    )

plt.title('Churn Rate by Tenure Group')
plt.xlabel('Tenure Group')
plt.ylabel('Churn Rate')
plt.ylim(0, group_stats['Churn_Rate'].max() * 1.1)
plt.tight_layout()
plt.show()
```



```python
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 5))
bars = sns.barplot(x='tenure_group', y='monthlycharges', data=avg_monthly, hue='tenure_group', palette='Set2', legend=False)

# Add value annotations on each bar
for bar in bars.patches:
    height = bar.get_height()
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        height + 1,
        f'{height:.2f}',
        ha='center',
        va='bottom',
        fontsize=10,
        fontweight='bold'
    )

plt.title('Average Monthly Charges Across Tenure Groups')
plt.xlabel('Tenure Group')
plt.ylabel('Average Monthly Charges (USD)')
plt.ylim(0, avg_monthly['monthlycharges'].max() + 10)
plt.tight_layout()
plt.show()
```



TASK 5: ADVANCED ANALYSIS

- Perform deeper analysis by grouping customers by tenure to compute statistics for charges and churn.
- Analyze churn rates by demographics (e.g., gender, senior citizen status).
- Payment methods and contract types. Visualize trends over time (if applicable) or lifecycle stages to identify patterns.

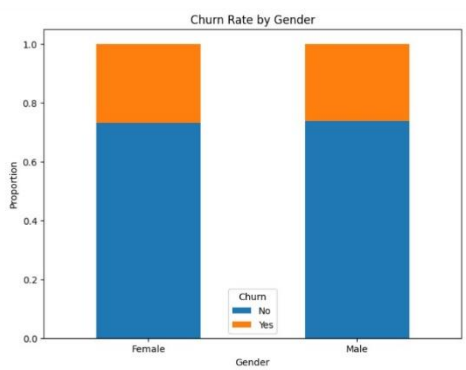
```
[19] # Create tenure groups
df['tenure_group'] = pd.cut(df['tenure'], bins=[0, 12, 36, 72], labels=['0-12', '13-36', '37+'])

# Group by tenure group and compute average charges and churn rate
tenure_analysis = df.groupby('tenure_group', observed=False).agg({
    'monthlycharges': 'mean',
    'totalcharges': 'mean',
    'churn': 'mean'  # churn rate
}).reset_index()

print(tenure_analysis)
```

```
  tenure_group  monthlycharges  totalcharges     churn
0         0-12       56.172023    276.621563  0.476782
1        13-36       63.248195   1513.541756  0.255388
2          37+       72.008730   4213.723192  0.119294
```

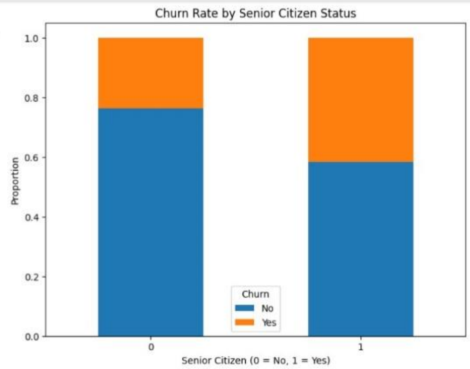
```
# Churn rate by gender
gender_churn = df.groupby('gender')['churn'].value_counts(normalize=True).unstack()
print(gender_churn)

# Churn rate by senior citizen status
senior_churn = df.groupby('seniorcitizen')['churn'].value_counts(normalize=True).unstack()
print(senior_churn)
```

```
churn          0         1
gender
Female  0.730791  0.269209
Male    0.738397  0.261603
churn                 0         1
seniorcitizen
0              0.763938  0.236062
```

```
[23] # Churn rate by gender
gender_churn.plot(kind='bar', stacked=True, figsize=(8, 6))
plt.title('Churn Rate by Gender')
plt.xlabel('Gender')
plt.ylabel('Proportion')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['No', 'Yes'])
plt.show()
```

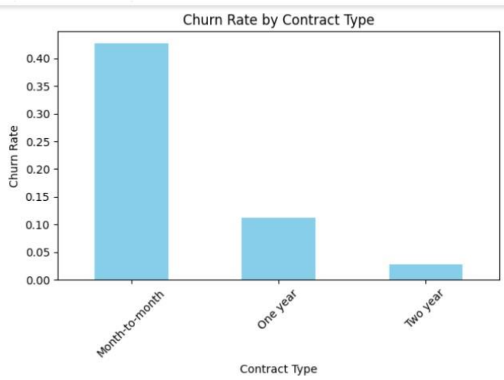


```python
# Churn rate by senior citizen status
senior_churn.plot(kind='bar', stacked=True, figsize=(8, 6))
plt.title('Churn Rate by Senior Citizen Status')
plt.xlabel('Senior Citizen (0 = No, 1 = Yes)')
plt.ylabel('Proportion')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['No', 'Yes'])
plt.show()
```




```python
# Churn rate by payment method
payment_churn = df.groupby('paymentmethod')['churn'].value_counts(normalize=True).unstack()
print(payment_churn)

# Churn rate by contract type (barplot)
import seaborn as sns
import matplotlib.pyplot as plt

contract_churn = df.groupby('contract')['churn'].value_counts(normalize=True).unstack()
contract_churn[1].plot(kind='bar', color='skyblue')

plt.title('Churn Rate by Contract Type')
plt.ylabel('Churn Rate')
plt.xlabel('Contract Type')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
churn                             0         1
paymentmethod
Bank transfer (automatic)  0.832902  0.167098
Credit card (automatic)    0.847569  0.152431
Electronic check           0.547146  0.452854
Mailed check               0.808933  0.191067
```



```
# Line plot of churn rate over tenure
tenure_churn = df.groupby('tenure')['churn'].apply(lambda x: (x == 1).mean())

plt.figure(figsize=(10,5))
tenure_churn.plot()
plt.title('Churn Rate Across Tenure (Months)')
plt.xlabel('Tenure (Months)')
plt.ylabel('Churn Rate')
plt.grid(True)
plt.tight_layout()
plt.show()
```