In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


dataset = pd.read_csv("C:/Users/sunda/Desktop/PORTFOLIO/TECH _CUSTOMER CHURN/telco.csv")
```

In [2]:
```
pd.set_option('display.max_columns', None)
```

In [3]:
```
dataset.head(10)
```

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Inte
0	7590-VHVEG	Female	0	Yes	No	1	No	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	
3	7795-CFOCW	Male	0	No	No	45	No	No	
4	9237-HQITU	Female	0	No	No	2	Yes	No	
5	9305-CDSKC	Female	0	No	No	8	Yes	Yes	
6	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	
7	6713-OKOMC	Female	0	No	No	10	No	No	
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	
9	6388-TABGU	Male	0	No	Yes	62	Yes	No	

In [4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7032 entries, 0 to 7031
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7032 non-null   object 
 1   gender            7032 non-null   object 
 2   SeniorCitizen     7032 non-null   int64  
 3   Partner           7032 non-null   object 
 4   Dependents        7032 non-null   object 
 5   tenure            7032 non-null   int64  
 6   PhoneService      7032 non-null   object 
 7   MultipleLines     7032 non-null   object 
 8   InternetService   7032 non-null   object 
 9   OnlineSecurity    7032 non-null   object 
 10  OnlineBackup      7032 non-null   object 
 11  DeviceProtection  7032 non-null   object 
 12  TechSupport       7032 non-null   object 
 13  StreamingTV       7032 non-null   object 
 14  StreamingMovies   7032 non-null   object 
 15  Contract          7032 non-null   object 
 16  PaperlessBilling  7032 non-null   object 
 17  PaymentMethod     7032 non-null   object 
 18  MonthlyCharges    7032 non-null   float64
 19  TotalCharges      7032 non-null   float64
 20  Churn             7032 non-null   object 
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

In [5]:
```
dataset.describe()
```

Out[5]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7032.000000	7032.000000	7032.000000	7032.000000
mean	0.162400	32.421786	64.798208	2283.300441
std	0.368844	24.545260	30.085974	2266.771362
min	0.000000	1.000000	18.250000	18.800000
25%	0.000000	9.000000	35.587500	401.450000
50%	0.000000	29.000000	70.350000	1397.475000
75%	0.000000	55.000000	89.862500	3794.737500
max	1.000000	72.000000	118.750000	8684.800000

In [6]:
```
dataset.drop_duplicates(inplace =True)
```

In [7]:
```
dataset.count()
```

Out[7]:
```
customerID          7032
gender              7032
SeniorCitizen       7032
Partner             7032
Dependents          7032
tenure              7032
PhoneService        7032
MultipleLines       7032
InternetService     7032
OnlineSecurity      7032
OnlineBackup        7032
DeviceProtection    7032
TechSupport         7032
StreamingTV         7032
StreamingMovies     7032
Contract            7032
PaperlessBilling    7032
PaymentMethod       7032
MonthlyCharges      7032
TotalCharges        7032
Churn               7032
dtype: int64
```

In [8]:
```
dataset.isnull().sum()
```

Out[8]:
```
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 [9]:
```
dataset.columns
```

Out[9]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [10]:
```
dataset.isnull().values.any()
```

Out[10]:
```
np.False_
```

In [11]:
```
##FOUND OUT
##astype(int)
##dataset[newcolumn ] = np.where(dataset[oldcolumn] == yes,1,0)
```

In [12]:
```
#[CALCULATED COLUMN]

## CUSTOMER LIFETIME VALUE = tenure * monthly


dataset['Customer-lifetime-value'] = dataset['MonthlyCharges'] * dataset['tenure']
print('Customer-lifetime-value')
```

```
Customer-lifetime-value
```

In [13]:
```
## TOTAL SERVICES USED BY CUSTOMERS

#CHNANGING TEXT COLUMN TO INT[YES =1, NO= 0]
##DROPPED OLDER COLUMN [INT SERVICE]

#count all services yes to 1 [then sum all of them up ]
#create a new column to add up all 1s as total services used by each customers THEN PRINT CHU
```

In [17]:
```
## REPLACING COLUMNS FROM YES TO 1

Conversion = ['PhoneService', 'InternetService', 'MultipleLines', 'OnlineSecurity', 'OnlineBa
            'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']

dataset[Conversion] = dataset[Conversion].replace({'Yes':1, 'No':0})
```

In [18]:
```
##NUMBER OF SERVICES USED BY CUSTOMER

Conversion = ['PhoneService', 'InternetService', 'MultipleLines', 'OnlineSecurity', 'OnlineBa
            'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']

dataset['Total_Service_Used'] = dataset[Conversion].sum(axis=1)
print(dataset[['customerID','Total_Service_Used', 'Churn']].head(10))
```

```
   customerID  Total_Service_Used Churn
0  7590-VHVEG                   2    No
1  5575-GNVDE                   4    No
2  3668-QPYBK                   4   Yes
3  7795-CFOCW                   4    No
4  9237-HQITU                   2   Yes
5  9305-CDSKC                   6   Yes
6  1452-KIOVK                   5    No
7  6713-OKOMC                   2    No
8  7892-POOKP                   7   Yes
9  6388-TABGU                   4    No
```

In [33]:
```
## RISK SCORE ANALYSIS/DISTRIBUTION

dataset['Risk_score'] = 0
dataset.loc[dataset['tenure'] < 12, 'Risk_score'] += 30
dataset.loc[dataset['Contract'] == 'Month-to-Month', 'Risk_score'] += 25
dataset.loc[dataset['MonthlyCharges'] > 70, 'Risk_score'] += 20
dataset.loc[dataset['Total_Service_Used'] <3, 'Risk_score'] +=15
dataset.loc[dataset['TechSupport'] == 0, 'Risk_score'] += 10



dataset['Risk_Category'] = 'Low Risk'
dataset.loc[dataset['Risk_score'] >= 40, 'Risk_Category'] = 'Medium Risk'
dataset.loc[dataset['Risk_score'] >= 70, 'Risk_Category'] = 'High Risk'
```

In [34]:
```
print(dataset[['Risk_score', 'Risk_Category', 'Churn']].head(20))
```

```
    Risk_score Risk_Category Churn
0           55   Medium Risk    No
1           10      Low Risk    No
2           40   Medium Risk   Yes
3            0      Low Risk    No
4           75     High Risk   Yes
5           60   Medium Risk   Yes
6           30      Low Risk    No
7           55   Medium Risk    No
8           20      Low Risk   Yes
9           10      Low Risk    No
10          10      Low Risk    No
11          25      Low Risk    No
12          30      Low Risk    No
13          30      Low Risk   Yes
14          20      Low Risk    No
15          20      Low Risk    No
16          25      Low Risk    No
17          30      Low Risk    No
18          30      Low Risk   Yes
19          30      Low Risk    No
```

In [35]:
```
print(dataset['Risk_Category'].value_counts())
```

```
Risk_Category
Low Risk       5081
Medium Risk    1861
High Risk        90
Name: count, dtype: int64
```

In [36]:
```
#risk_analysis = dataset.groupby('Risk_Category')['Churn'].apply(lambda x: (x == 'Yes').sum()
#print(risk_analysis)

#or

dataset.groupby('Risk_Category')['Churn'].apply(lambda x: (x == 'Yes').sum() / len(x) * 100).
```

Out[36]:
```
Risk_Category
High Risk      61.11
Low Risk       17.63
Medium Risk    49.33
Name: Churn, dtype: float64
```

In [37]:
```
"I created a risk scoring model that assigns points based on 5 key churn indicators.
Customers scoring 70+ are flagged as High Risk - they have a 41% churn rate
vs
39% for Low Risk customers and keeping an eye on 19% for medium risk customers.
The company can now proactively target these
31 High Risk customers
and
2599 medium risk customers
with retention offers before they leave."
```

```
  Cell In[37], line 1
    "I created a risk scoring model that assigns points based on 5 key churn indicators.
    ^
SyntaxError: unterminated string literal (detected at line 1)
```

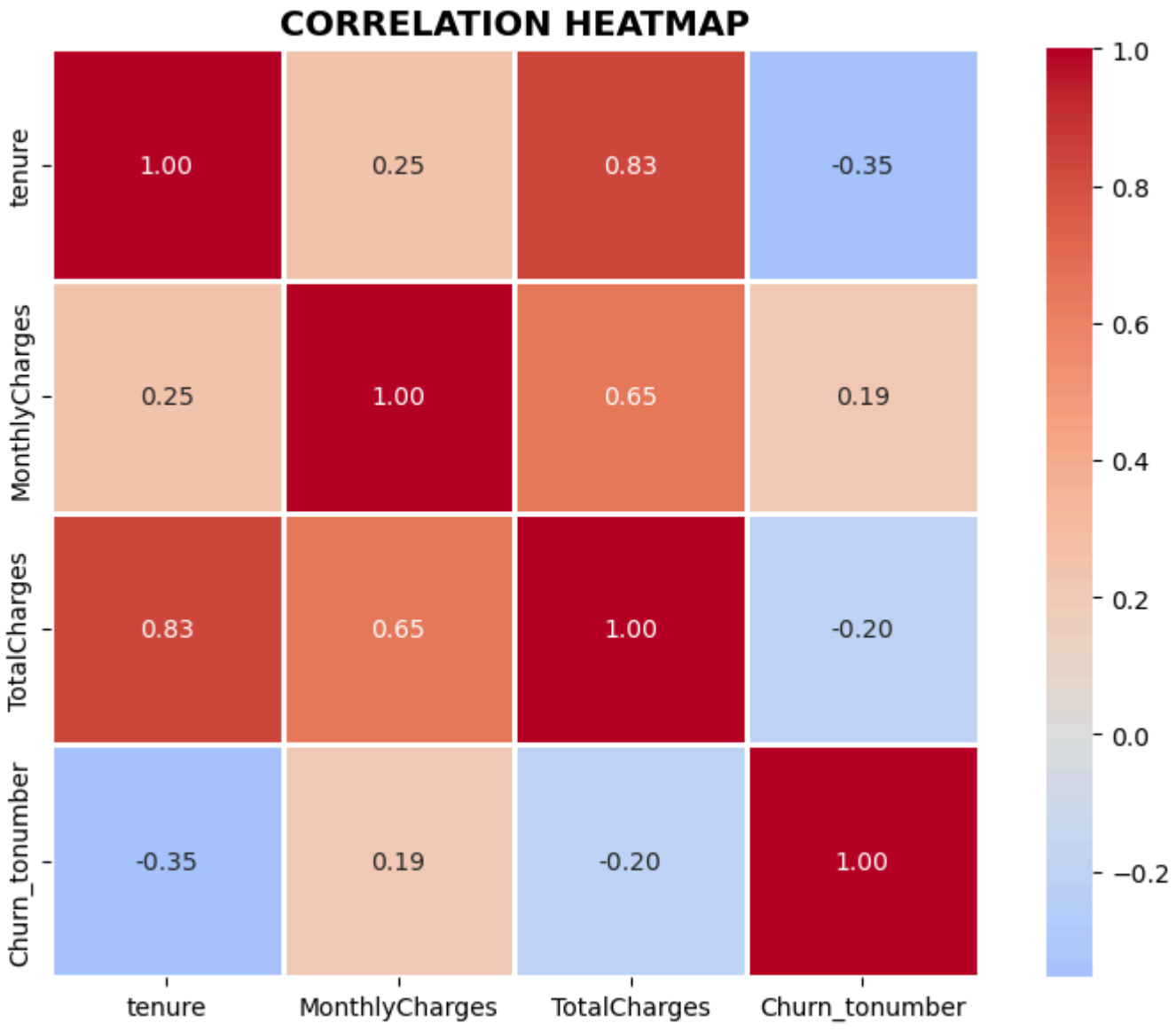
In [38]:
```
#STATISTICAL ANALYSIS IN PYTHON
#CORRELATION ANALYSIS


dataset['TotalCharges'] = pd.to_numeric(dataset['TotalCharges'], errors = 'coerce')
dataset['TotalCharges'] = dataset['TotalCharges'].fillna(0)

dataset['Churn_tonumber'] = dataset['Churn'].replace({'Yes': 1, 'No': 0})
```

```
number_cols = ['tenure', 'MonthlyCharges', 'TotalCharges', 'Churn_tonumber']

correlation = dataset[number_cols].corr()
print(correlation.round(2))
```

```
                tenure  MonthlyCharges  TotalCharges  Churn_tonumber
tenure            1.00            0.25          0.83           -0.35
MonthlyCharges    0.25            1.00          0.65            0.19
TotalCharges      0.83            0.65          1.00           -0.20
Churn_tonumber   -0.35            0.19         -0.20            1.00
```

```
C:\Users\sunda\AppData\Local\Temp\ipykernel_3700\1970231681.py:8: FutureWarning: Downcasting b
ehavior in `replace` is deprecated and will be removed in a future version. To retain the old
behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavio
r, set `pd.set_option('future.no_silent_downcasting', True)`
  dataset['Churn_tonumber'] = dataset['Churn'].replace({'Yes': 1, 'No': 0})
```

In [39]:
```
plt.figure(figsize=(8,6))
sns.heatmap(correlation, annot=True, cmap='coolwarm',center=0,square =True, linewidths =1, fm
plt.title('CORRELATION HEATMAP', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.savefig('correlation_heatmap.png')
plt.show()

print('Heatmap saved')
```



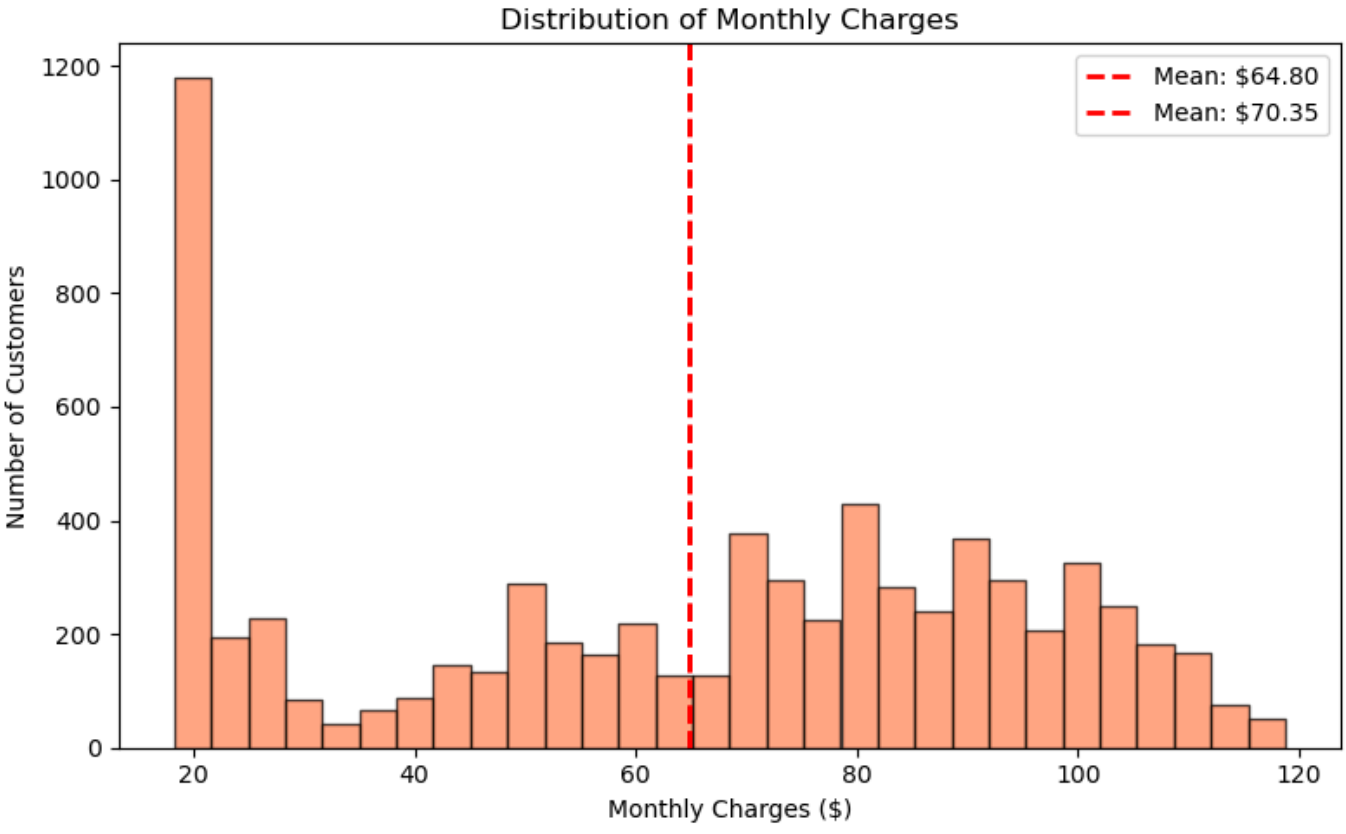

```
Heatmap saved
```

In [40]:
```
WHAT EACH PARAMETER DOES:
ParameterWhat It Doesannot=TrueShows the numbers inside boxescmap='coolwarm'Red = positive, B

WHAT YOU'LL SEE:
```

```
A colored grid with numbers showing relationships ✅
Red boxes = positive correlation
Blue boxes = negative correlation
White boxes = no correlation
```

```
  Cell In[40], line 2
    ParameterWhat It Doesannot=TrueShows the numbers inside boxescmap='coolwarm'Red = positiv
e, Blue = negativecenter=0Makes 0 white (neutral)fmt='.2f'Shows 2 decimals (0.35 not 0.353456)
square=TrueMakes boxes perfect squareslinewidths=1Adds borders between boxes

    ^
SyntaxError: invalid decimal literal
```

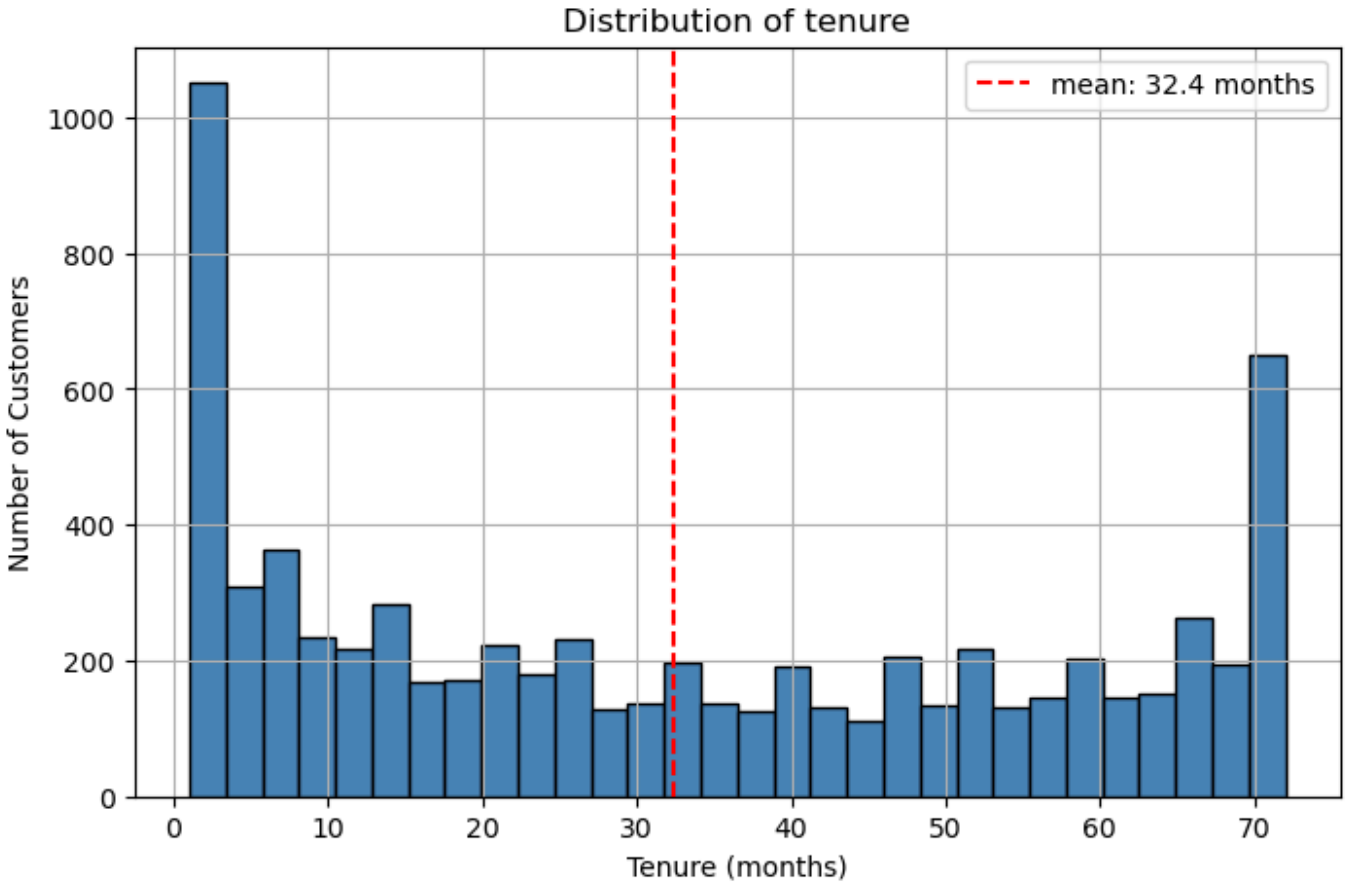
In [41]:
```
                                        #DISTRIBUTIONS ANALYSIS

# Distribution of Monthly Charges
plt.figure(figsize=(8, 5))
plt.hist(dataset['MonthlyCharges'], bins=30, color='coral', edgecolor='black', alpha=0.7)

#mean line
plt.axvline(dataset['MonthlyCharges'].mean(), color='red', linestyle='--',linewidth=2,
            label=f'Mean: ${dataset["MonthlyCharges"].mean():.2f}')

#median line
plt.axvline(dataset['MonthlyCharges'].mean(), color='red', linestyle='--',linewidth=2,
            label=f'Mean: ${dataset["MonthlyCharges"].median():.2f}')

plt.title('Distribution of Monthly Charges')
plt.xlabel('Monthly Charges ($)')
plt.ylabel('Number of Customers')
plt.tight_layout()
plt.savefig('monthly_charges_distribution.png')
plt.legend()
plt.show()
print('chart successfully saves')
```



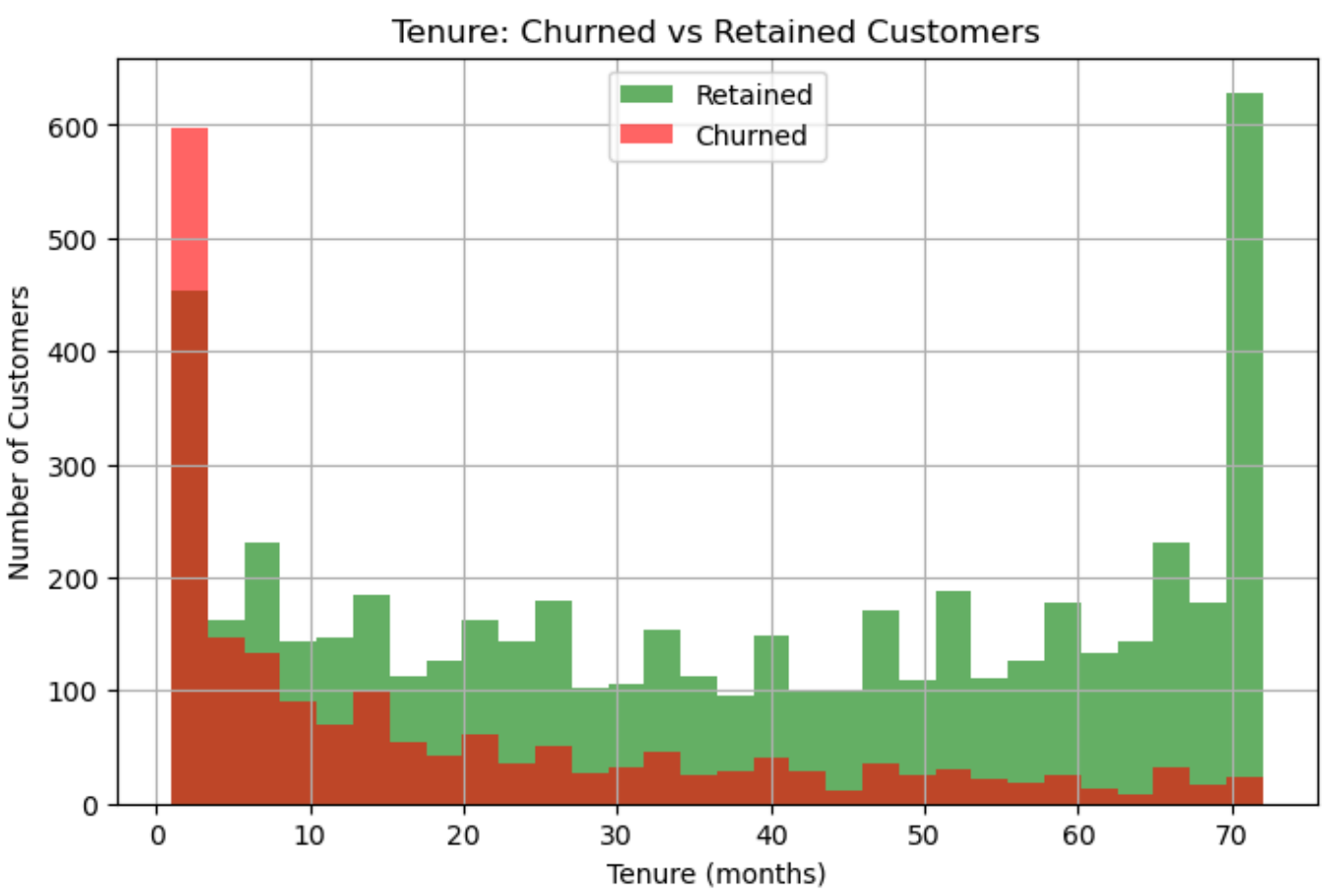

```
chart successfully saves
```

In [42]:
```
# Distribution of Tenure
plt.figure(figsize = (8, 5)), dataset['tenure'].hist(bins = 30, color='steelblue', edgecolor=
plt.title('Distribution of tenure')
```

```
plt.xlabel('Tenure (months)')
plt.ylabel('Number of Customers')
plt.axvline(dataset['tenure'].mean(), color='red', linestyle='--',
            label=f'mean: {dataset['tenure'].mean():.1f} months')
plt.legend()
plt.savefig('Distribution of tenure.png')
plt.show()
```

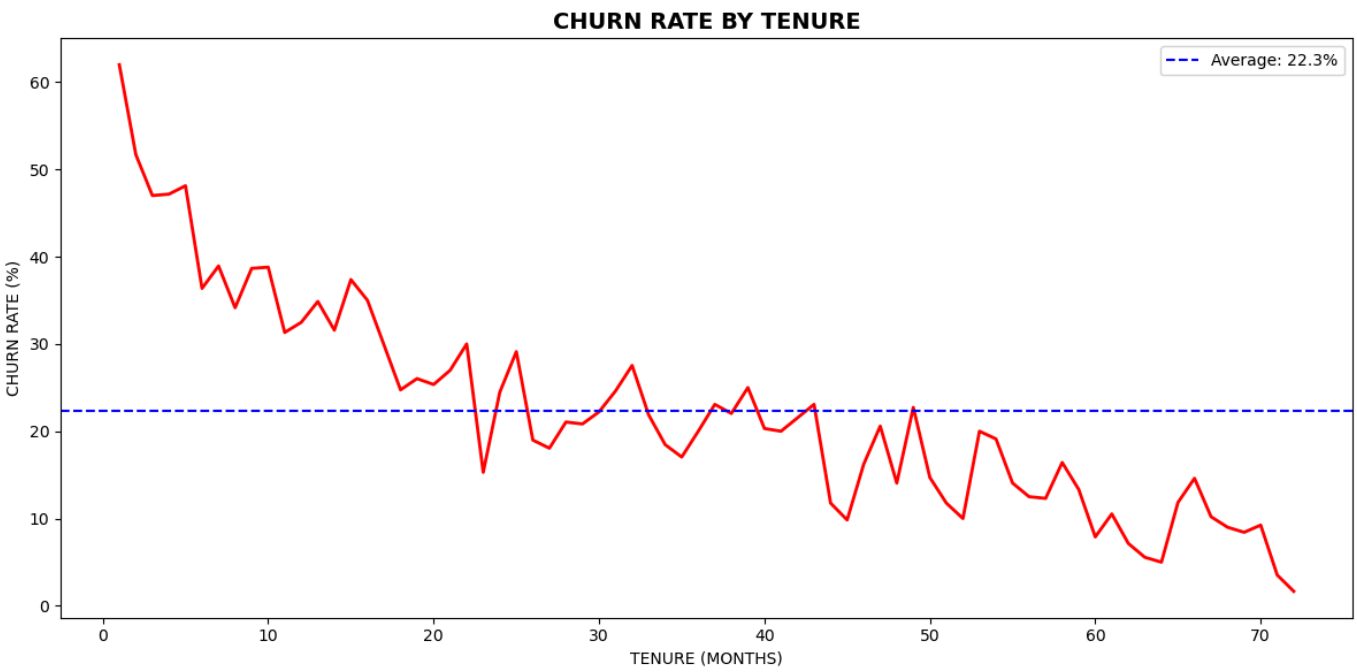


In [43]:
```
# Compare Tenure: Churned vs Retained
plt.figure(figsize=(8, 5))
dataset[dataset['Churn'] == 'No']['tenure'].hist(bins=30, alpha=0.6,
                                        color='green', label='Retained')
dataset[dataset['Churn'] == 'Yes']['tenure'].hist(bins=30, alpha=0.6,
                                         color='red', label='Churned')
plt.title('Tenure: Churned vs Retained Customers')
plt.xlabel('Tenure (months)')
plt.ylabel('Number of Customers')
plt.savefig('churned vs retained customers.png')
plt.legend()
plt.show()
```



In [44]:

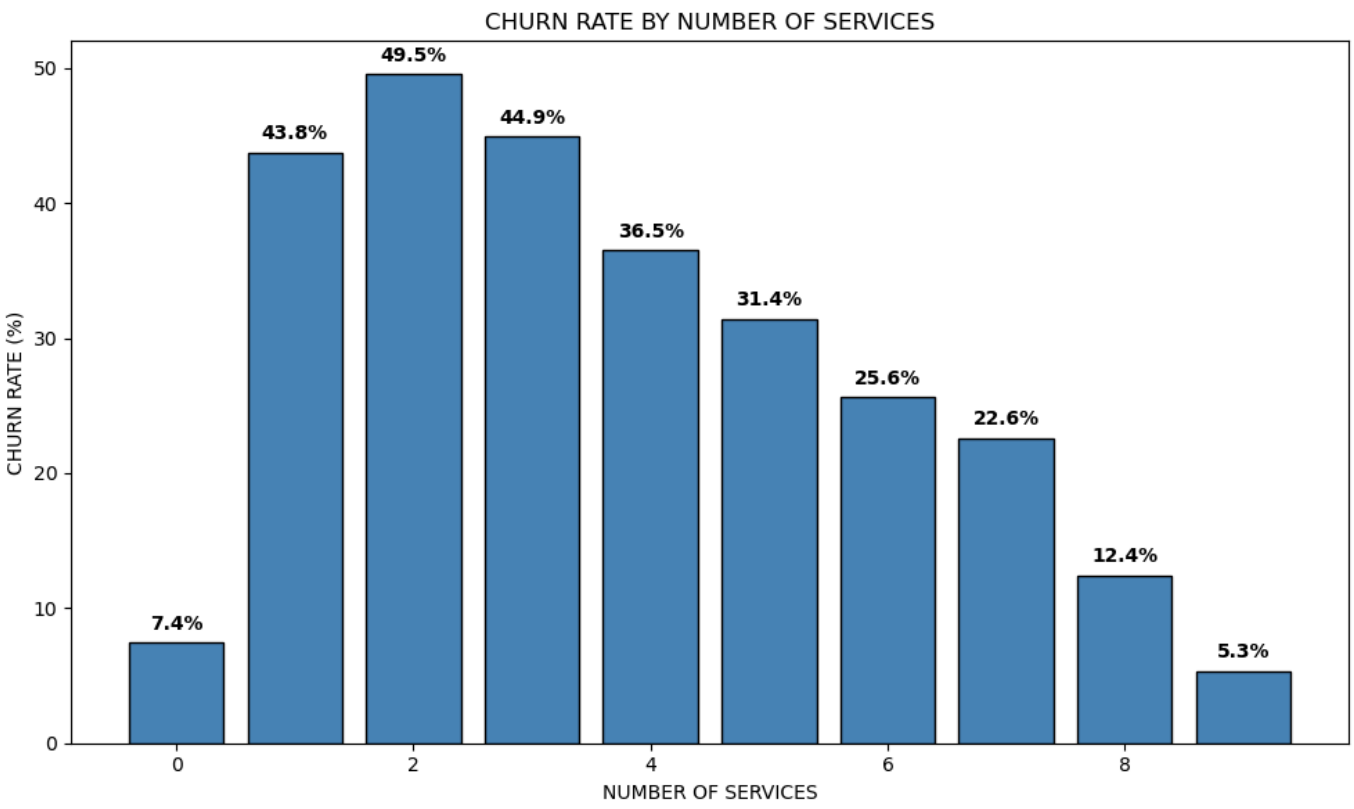
```
                                                    #VISUALISATIONS
                                                    #VISUALISATIONS

#CHURN RATE BY TENURE
churn_percentage = dataset.groupby('tenure')['Churn_tonumber'].mean() * 100
plt.figure(figsize=(12,6))

plt.plot(churn_percentage.index,churn_percentage.values,color='red', linewidth=2)
plt.title('CHURN RATE BY TENURE', fontsize=14, fontweight='bold')
plt.xlabel('TENURE (MONTHS)')
plt.ylabel('CHURN RATE (%)')

plt.axhline(churn_percentage.mean(), color='blue', linestyle='--', label=f'Average: {churn_pe


plt.legend()
plt.tight_layout()
plt.savefig('churn_by_tenure.png')
plt.show()
print('chart saved!')

print(churn_percentage.round(2))
```




```
chart saved!
tenure
1     61.99
2     51.68
3     47.00
4     47.16
5     48.12
      ...
68     9.00
69     8.42
70     9.24
71     3.53
72     1.66
Name: Churn_tonumber, Length: 72, dtype: float64
```

In [45]:

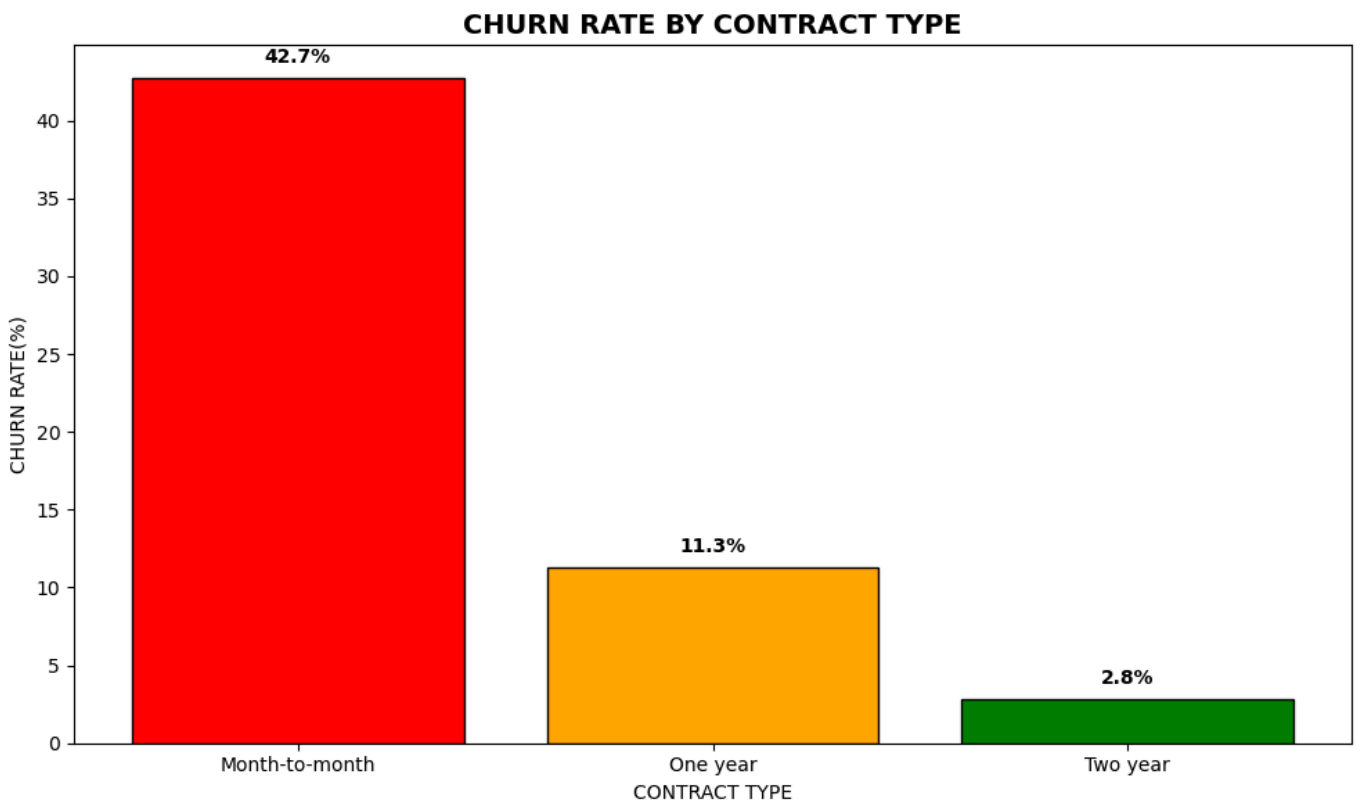
```python
#SERVICES VS CHURN(BAR-CHART)

Churn_by_services = dataset.groupby('Total_Service_Used')['Churn_tonumber'].mean() * 100
plt.figure(figsize=(10, 6))
plt.bar(Churn_by_services.index, Churn_by_services.values, color='steelblue', edgecolor='blac
plt.title('CHURN RATE BY NUMBER OF SERVICES')
plt.xlabel('NUMBER OF SERVICES')
plt.ylabel('CHURN RATE (%)')

for i, v in enumerate(Churn_by_services.values):
    plt.text(i, v + 1, f'{v:.1f}%', ha='center', fontweight='bold')
    plt.tight_layout()
    plt.savefig('services_vs_churn.png')

plt.show()

print('chart saved!')



#customers that used less services churns the most.
```




```
chart saved!
```

In [46]:

```python
#Contract type comparison (grouped bars)


Churn_by_contract = dataset.groupby('Contract')['Churn'].apply(
    lambda x: (x =='Yes').sum()/ len(x) * 100)

#or
##Churn_by_contract = dataset.groupby('Contract')['Churn_tonumber'].mean() * 100

#DRAWING GROUPED BAR CHART
plt.figure(figsize=(10,6))
plt.bar(Churn_by_contract.index, Churn_by_contract.values,
       color=['red','orange','green'], edgecolor='black')
plt.title('CHURN RATE BY CONTRACT TYPE',fontsize = 14,fontweight= 'bold')
plt.xlabel('CONTRACT TYPE')
plt.ylabel('CHURN RATE(%)')

#ADDING PERCENTAGES ON BARS
for i, v in enumerate(Churn_by_contract.values):
       plt.text(i, v + 1, f'{v:.1f}%', ha ='center', fontweight='bold')
plt.tight_layout()
plt.savefig('contract_vs_churn.png')
plt.show()
print('chart_saved')
```




```
chart_saved
```

In [47]:
```
dataset.to_csv('Cleaned telco_python.csv', index=False)
```

In [48]:
```
dataset.head()
```

Out[48]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Inte
0	7590-VHVEG	Female	0	Yes	No	1	0	0	
1	5575-GNVDE	Male	0	No	No	34	1	0	
2	3668-QPYBK	Male	0	No	No	2	1	0	
3	7795-CFOCW	Male	0	No	No	45	0	0	
4	9237-HQITU	Female	0	No	No	2	1	0	

In [ ]:

In [ ]:

In [ ]: