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

#Reading the data
customers_df= pd.read_csv("customers.csv")
customers_df.head()
```

Out[1]:

	CustomerID	FirstName	LastName	Email	PhoneNumber
0	1	John	Doe	john.doe@example.com	123-456-7890
1	2	Jane	Smith	jane.smith@example.com	098-765-4321
2	3	Alice	Johnson	alice.j@example.com	567-890-1234
3	4	Bob	Brown	bob.brown@example.com	234-567-8901
4	5	Charlie	Davis	charlie.d@example.com	345-678-9012

In [2]:
```
customers_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   CustomerID   300 non-null    int64 
 1   FirstName    299 non-null    object
 2   LastName     299 non-null    object
 3   Email        298 non-null    object
 4   PhoneNumber  298 non-null    object
 5   JoinDate     298 non-null    object
 6   Status       299 non-null    object
 7   Region       299 non-null    object
dtypes: int64(1), object(7)
memory usage: 18.9+ KB
```

In [3]:
```
customers_df.describe()
```

Out[3]:

	CustomerID
count	300.000000
mean	150.500000
std	86.746758
min	1.000000
25%	75.750000
50%	150.500000
75%	225.250000
max	300.000000

In [14]:
```
churn_df= pd.read_csv("churn.csv")
churn_df.head()
```

Out[14]:

	ChurnID	CustomerID	ChurnDate	Reason
0	1	22	02-01-2024	Poor Customer Service
1	2	56	06-06-2024	High Prices
2	3	45	10-10-2024	Product Quality Issues
3	4	54	12-12-2024	Lack of Product Features
4	5	82	14-02-2024	Competitor Offerings

In [9]:
```
Subscriptions_df= pd.read_csv("Subscriptions.csv")
Subscriptions_df.head()
```

Out[9]:

	SubscriptionID	CustomerID	StartDate	EndDate	PlanType
0	1	1	10-01-2022	09-01-2023	Annual
1	2	2	15-12-2021	14-12-2022	Monthly
2	3	3	20-03-2020	19-03-2021	Annual
3	4	4	25-06-2019	24-06-2020	Annual
4	5	5	14-07-2021	13-07-2022	Monthly

In [12]:
```
Transactions_Data= pd.read_csv("Transactions.csv")
Transactions_Data.head()
```

Out[12]:

	TransactionID	CustomerID	TransactionDate	Amount	TransactionType
0	1	15	01-01-2024	32	Purchase
1	2	22	02-01-2024	52	Refund
2	3	43	03-01-2024	165	Purchase
3	4	87	04-01-2024	134	Purchase
4	5	34	05-01-2024	158	Purchase

In [16]:
```
#1. Print first 5 customers FirstNames and LastNames

Customer_names= customers_df[['FirstName','LastName']]
Customer_names.head()
```

Out[16]:

	FirstName	LastName
0	John	Doe
1	Jane	Smith
2	Alice	Johnson
3	Bob	Brown
4	Charlie	Davis

In [23]:
```
#2. Find out customers from specific Region (North America)

Customer_Region=customers_df[customers_df['Region']== 'North America']
Customer_Region.head()
print(Customer_Region)
```

```
     CustomerID FirstName LastName                    Email   PhoneNumber
\
0             1      John      Doe       john.doe@example.com  123-456-7890
3             4       Bob    Brown      bob.brown@example.com  234-567-8901
6             7       Eva   Harris          eva.h@example.com  567-890-1234
9            10      Hank   Wilson        hank.w@example.com  890-123-4567
12           13     Kelly    Lewis       kelly.l@example.com  123-456-7890
..          ...       ...      ...                       ...           ...
287         288    Rachel   Harris  rachel.harris@example.com  678-901-2345
290         291     Lucas   Taylor   lucas.taylor@example.com  901-234-5678
293         294    Sophia   Morris  sophia.morris@example.com  234-567-8901
296         297      Liam   Wilson    liam.wilson@example.com  567-890-1234
299         300     Lucas      Lee      lucas.lee@example.com  890-123-4567

       JoinDate    Status         Region
0    10-01-2022    Active  North America
3    25-06-2019  Inactive  North America
6    22-12-2020  Inactive  North America
9    27-05-2020    Active  North America
12   15-04-2018    Active  North America
..          ...       ...            ...
287  05-03-2019  Inactive  North America
290  19-08-2019    Active  North America
293  14-12-2019  Inactive  North America
296  10-08-2020    Active  North America
299  01-11-2021  Inactive  North America

[103 rows x 8 columns]
```

In [26]:
```
#3. Find customers in specific Regions (Europe or Asia)

Customer_region=customers_df[customers_df['Region'].isin(['Europe','Asia'])]
print(Customer_region)
```

```
     CustomerID FirstName LastName                    Email   PhoneNumber
\
1             2      Jane    Smith    jane.smith@example.com  098-765-4321
2             3     Alice  Johnson       alice.j@example.com  567-890-1234
4             5   Charlie    Davis     charlie.d@example.com  345-678-9012
5             6     Diana    Clark       diana.c@example.com  456-789-0123
7             8     Frank   Garcia       frank.g@example.com  678-901-2345
..          ...       ...      ...                       ...           ...
292         293      Jake    Young    jake.young@example.com  123-456-7890
294         295     Aiden    Davis   aiden.davis@example.com  345-678-9012
295         296     Emily    Brown   emily.brown@example.com  456-789-0123
297         298    Olivia    Perry  olivia.perry@example.com  678-901-2345
298         299      Ella  Johnson  ella.johnson@example.com  789-012-3456

       JoinDate    Status  Region
1    15-12-2021  Inactive  Europe
2    20-03-2020    Active    Asia
4    14-07-2021    Active  Europe
5    30-08-2018    Active    Asia
7    11-03-2019    Active  Europe
..          ...       ...     ...
292  29-05-2021    Active    Asia
294  03-11-2022    Active  Europe
295  19-03-2021  Inactive    Asia
297  12-06-2019  Inactive  Europe
298  18-05-2022    Active    Asia

[196 rows x 8 columns]
```

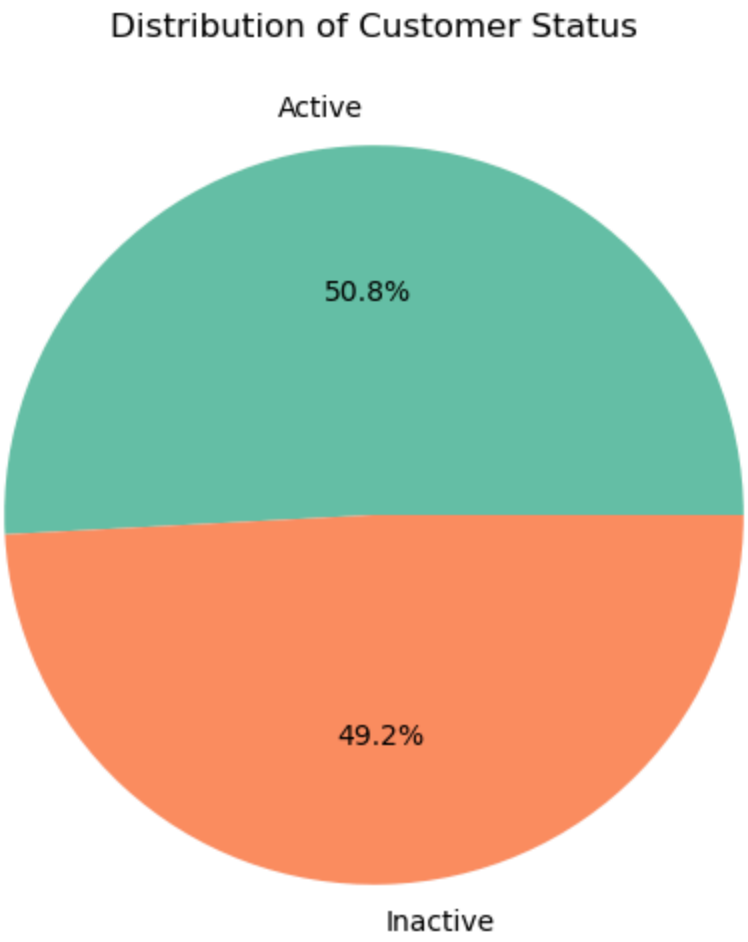
In [29]:

```python
#4. Find Customers with Active Status

status= customers_df[customers_df['Status']== 'Active']
status.head()
print(status)
```

```
     CustomerID FirstName LastName                    Email   PhoneNumber
\
0             1      John      Doe      john.doe@example.com  123-456-7890
2             3     Alice  Johnson       alice.j@example.com  567-890-1234
4             5   Charlie    Davis     charlie.d@example.com  345-678-9012
5             6     Diana    Clark       diana.c@example.com  456-789-0123
7             8     Frank   Garcia       frank.g@example.com  678-901-2345
..          ...       ...      ...                       ...           ...
290         291     Lucas   Taylor  lucas.taylor@example.com  901-234-5678
292         293      Jake    Young    jake.young@example.com  123-456-7890
294         295     Aiden    Davis   aiden.davis@example.com  345-678-9012
296         297      Liam   Wilson   liam.wilson@example.com  567-890-1234
298         299      Ella  Johnson  ella.johnson@example.com  789-012-3456

       JoinDate  Status         Region
0    10-01-2022  Active  North America
2    20-03-2020  Active           Asia
4    14-07-2021  Active         Europe
5    30-08-2018  Active           Asia
7    11-03-2019  Active         Europe
..          ...     ...            ...
290  19-08-2019  Active  North America
292  29-05-2021  Active           Asia
294  03-11-2022  Active         Europe
296  10-08-2020  Active  North America
298  18-05-2022  Active           Asia

[152 rows x 8 columns]
```

In [31]:

```python
#5. Visualization of Active & Inactive Customers

#Calculate the number of active / Inactive customers

status_count=customers_df['Status'].value_counts()

#Create a pie chart to show the distribution

plt.figure(figsize=(8,6))
plt.pie(status_count, labels=status_count.index, autopct='%1.1f%%', colors=[
plt.title('Distribution of Customer Status')
plt.show()
```

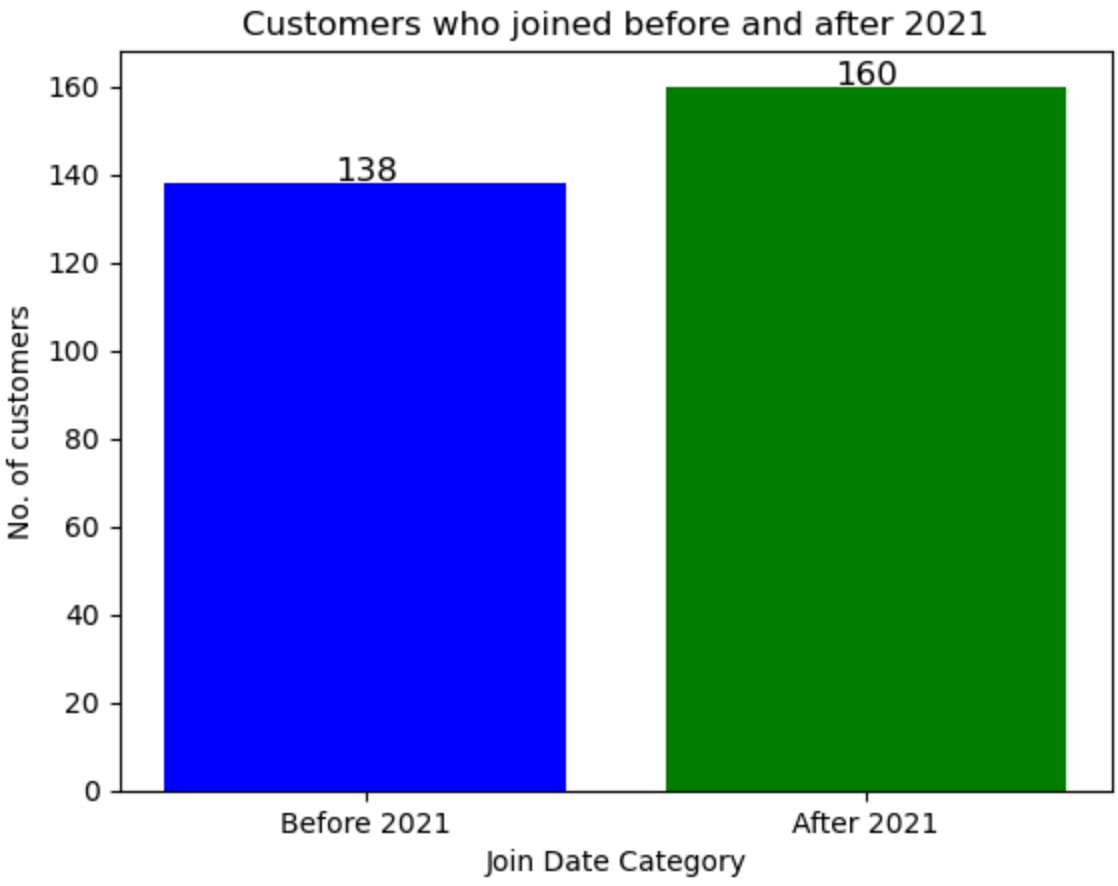


In [36]:
```
#6. List down the customers who joined after '2020-01-01'
#convert the 'JoinDate' column to date time format
customer_joined=customers_df[pd.to_datetime(customers_df['JoinDate'],format=
print(customer_joined)
```

```
     CustomerID FirstName LastName                    Email   PhoneNumber
\
0             1      John      Doe      john.doe@example.com  123-456-7890
1             2      Jane    Smith    jane.smith@example.com  098-765-4321
2             3     Alice  Johnson       alice.j@example.com  567-890-1234
4             5   Charlie    Davis     charlie.d@example.com  345-678-9012
6             7       Eva   Harris         eva.h@example.com  567-890-1234
..          ...       ...      ...                       ...           ...
294         295     Aiden    Davis   aiden.davis@example.com  345-678-9012
295         296     Emily    Brown   emily.brown@example.com  456-789-0123
296         297      Liam   Wilson   liam.wilson@example.com  567-890-1234
298         299      Ella  Johnson  ella.johnson@example.com  789-012-3456
299         300     Lucas      Lee     lucas.lee@example.com  890-123-4567

       JoinDate    Status         Region
0    10-01-2022    Active  North America
1    15-12-2021  Inactive         Europe
2    20-03-2020    Active           Asia
4    14-07-2021    Active         Europe
6    22-12-2020  Inactive  North America
..          ...       ...            ...
294  03-11-2022    Active         Europe
295  19-03-2021  Inactive           Asia
296  10-08-2020    Active  North America
298  18-05-2022    Active           Asia
299  01-11-2021  Inactive  North America

[228 rows x 8 columns]
```

In [39]:

```
#7. Create a visualisation(bar chart) who joined after and befoe 2021
 #converting JoinDate into DATE TIME FORMAT
customers_df['JoinDate']=pd.to_datetime(customers_df['JoinDate'],format='%d-

# Filtering the customers who joined before and after 2021

before_2021=customers_df[customers_df['JoinDate']<'2021-01-01']
after_2021=customers_df[customers_df['JoinDate']>'2021-01-01']

# Count the number of customers in each of the category
counts={'Before 2021': len(before_2021),'After 2021': len(after_2021)}

# create a bar chart
plt.bar(counts.keys(),counts.values(), color=['blue','green'])
plt.xlabel('Join Date Category')
plt.ylabel('No. of customers')
plt.title('Customers who joined before and after 2021')


#Add the exact numbers on the top of the bars
for i, (key,value) in enumerate(counts.items()):
    plt.text(i,value+0.5, str(value), ha='center', fontsize=12)
plt.show()

# Using for loop, the loop goes through each bar (Before 2021 and After 2021
#For each bar, it places the customer count(value) as a label slightly above
#counts.items(): This gives you both the key and the values from dictionary
```

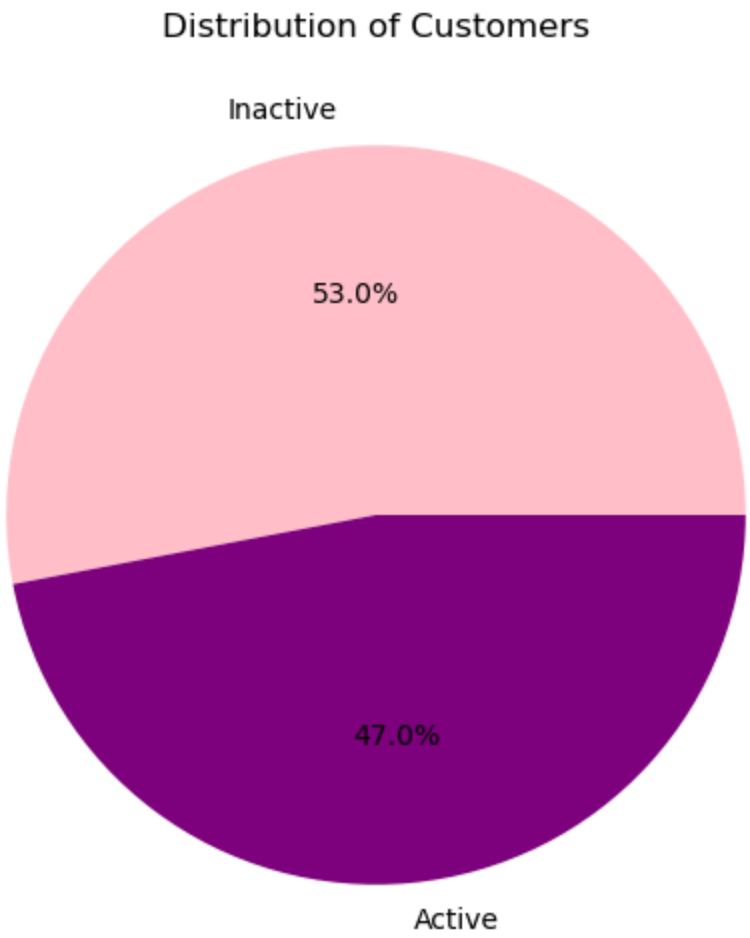


In [ ]:

In [44]:
```
#8. List down the active customers from Europe
Active_cx=customers_df[(customers_df['Region']=='Europe') & (customers_df['S
result=len(Active_cx)
print(result)
```

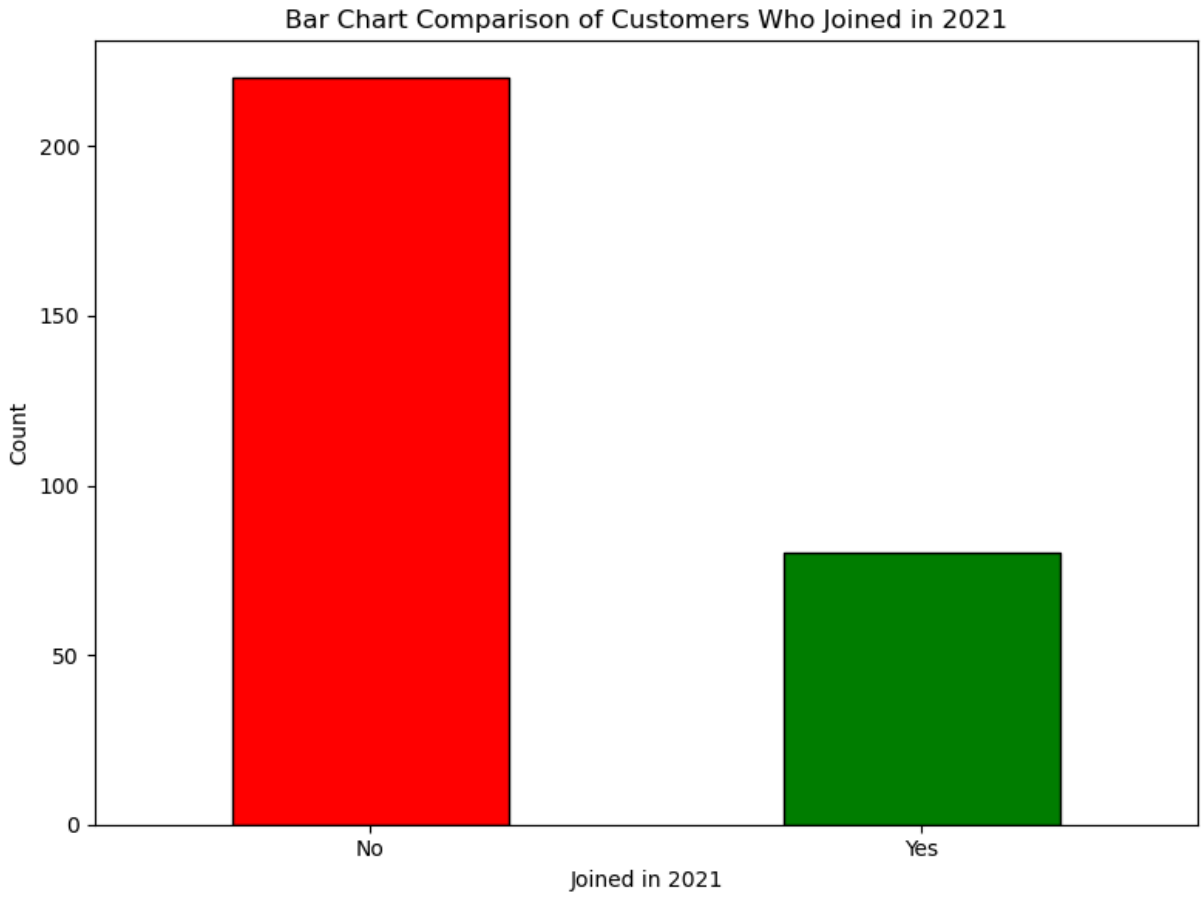
47

In [49]:
```
#Visualisation of active and Inactive Customers in Europe.
# Filtering the european customers
customers_europe=customers_df[customers_df['Region']=='Europe']

#Count the number of active and Inactive
status_counts= customers_europe['Status'].value_counts()

plt.figure(figsize=(8,6))
plt.pie(status_counts, labels=status_counts.index, autopct='%1.1f%%',colors=
plt.title('Distribution of Customers')
plt.show()
```



In [101…

```
customers_df['JoinDate']=pd.to_datetime(customers_df['JoinDate'],format='%d-
customers_df['Joined2021'] = pd.to_datetime(customers_df['JoinDate']).dt.yea

# Count the number of customers who are in yes or no
joined_2021_counts = customers_df['Joined2021'].value_counts()

# Create a bar chart
plt.figure(figsize=(8, 6))
joined_2021_counts.plot(kind='bar', color=['red', 'green'], edgecolor='black
plt.xlabel('Joined in 2021')
plt.ylabel('Count')
plt.title('Bar Chart Comparison of Customers Who Joined in 2021')
plt.xticks(ticks=[0, 1], labels=['No', 'Yes'], rotation=0)
plt.tight_layout()

plt.show()
```



In [95]:
```
customers_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   CustomerID   300 non-null    int64 
 1   FirstName    299 non-null    object
 2   LastName     299 non-null    object
 3   Email        298 non-null    object
 4   PhoneNumber  298 non-null    object
 5   JoinDate     298 non-null    object
 6   Status       299 non-null    object
 7   Region       299 non-null    object
dtypes: int64(1), object(7)
memory usage: 18.9+ KB
```

In [98]:
```
customers_df.describe()
```

Out[98]:

	CustomerID
count	300.000000
mean	150.500000
std	86.746758
min	1.000000
25%	75.750000
50%	150.500000
75%	225.250000
max	300.000000

In [102…

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#Reading the data
Transactions_df= pd.read_csv("Transactions.csv")
Transactions_df.head()
```

Out[102…

	TransactionID	CustomerID	TransactionDate	Amount	TransactionType
0	1	15	01-01-2024	32	Purchase
1	2	22	02-01-2024	52	Refund
2	3	43	03-01-2024	165	Purchase
3	4	87	04-01-2024	134	Purchase
4	5	34	05-01-2024	158	Purchase

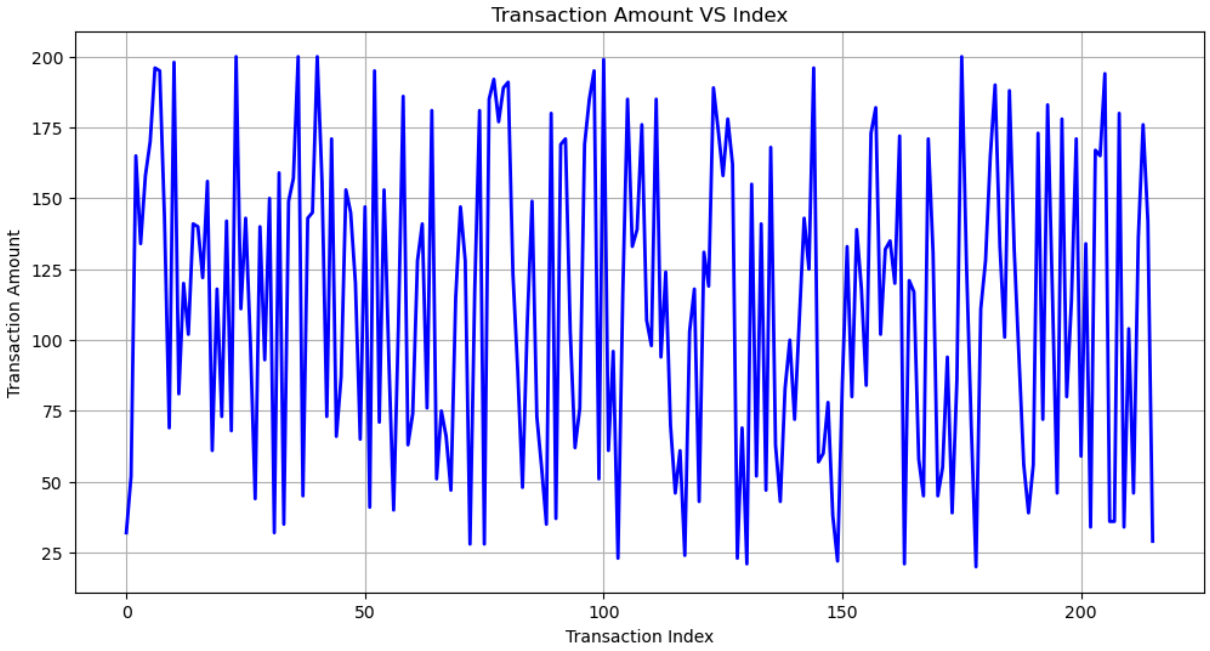
In [103…

```
#calculate the average transaction amount
Average_amount= Transactions_df['Amount'].mean()
print(f"Average Transaction Amount: {Average_amount}")
```

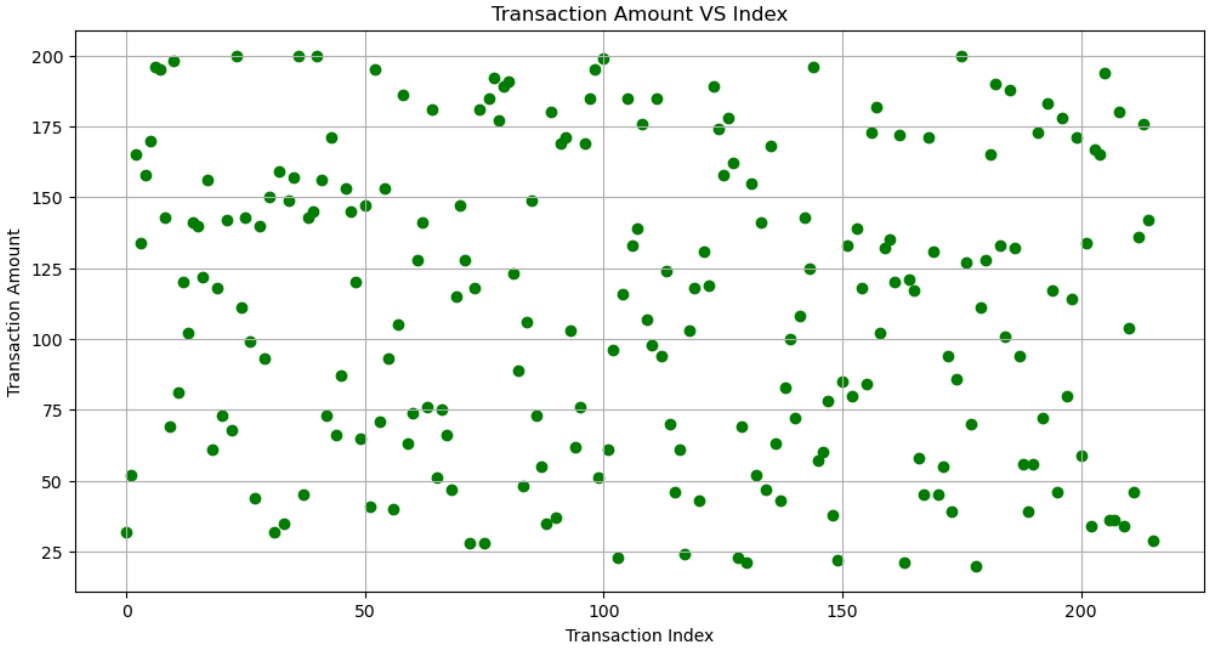
```
Average Transaction Amount: 112.19907407407408
```

In [111…

```
# Visualize / Analysis for Transaction Amount w.r.t to Index - Line graph
plt.figure(figsize=(12, 6))
plt.plot(Transactions_df.index, Transactions_df['Amount'],linestyle='-', lir

plt.title('Transaction Amount VS Index')
plt.xlabel('Transaction Index')
plt.ylabel('Transaction Amount')
plt.grid(True)
plt.show()
```



In [116…
```
# Visualize / Analysis for Transaction Amount w.r.t to Index - scatter plot
plt.figure(figsize=(12, 6))
plt.scatter(Transactions_df.index, Transactions_df['Amount'],color='Green')

plt.title('Transaction Amount VS Index')
plt.xlabel('Transaction Index')
plt.ylabel('Transaction Amount')
plt.grid(True)
plt.show()
```



In [118…
```
# Transactions which are more than $100
Max_amount= Transactions_df[Transactions_df['Amount']>100]
print(Max_amount)
```

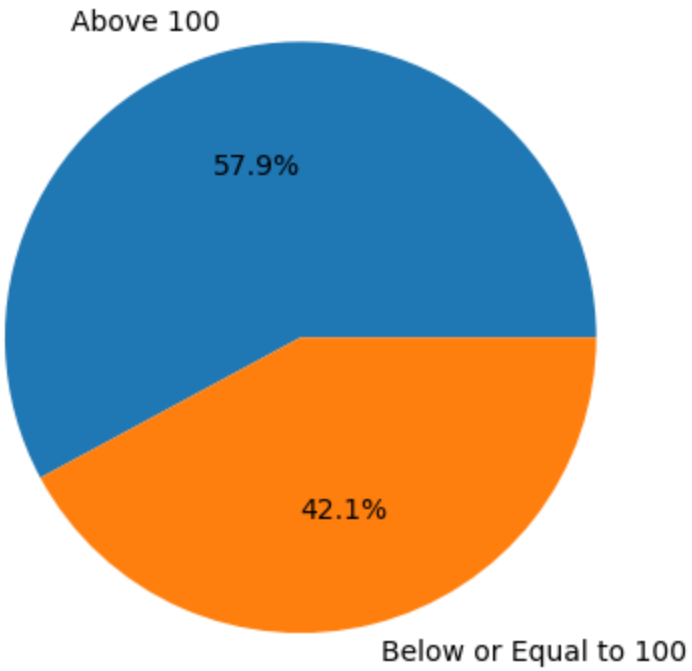
```
     TransactionID  CustomerID TransactionDate  Amount TransactionType
2                3          43      03-01-2024     165        Purchase
3                4          87      04-01-2024     134        Purchase
4                5          34      05-01-2024     158        Purchase
5                6          56      06-01-2024     170          Refund
6                7          23      07-01-2024     196        Purchase
..             ...         ...             ...     ...             ...
208            209         263      01-09-2024     180          Refund
210            211         265      03-09-2024     104        Purchase
212            213         267      05-09-2024     136          Refund
213            214         268      06-09-2024     176        Purchase
214            215         269      07-09-2024     142        Purchase

[125 rows x 5 columns]
```

In [121…

```
# Using a pie chart .. show me the % breakdown of  "Above $100 and Below / E
above_100= Transactions_df[Transactions_df['Amount']>100].shape[0]
Below_or_equal_100 = Transactions_df[Transactions_df['Amount']<=100].shape[0

labels= ['Above 100', 'Below or Equal to 100']
sizes= [above_100, Below_or_equal_100]

plt.pie(sizes, labels=labels, autopct='%1.1f%%')
```

Out[121…

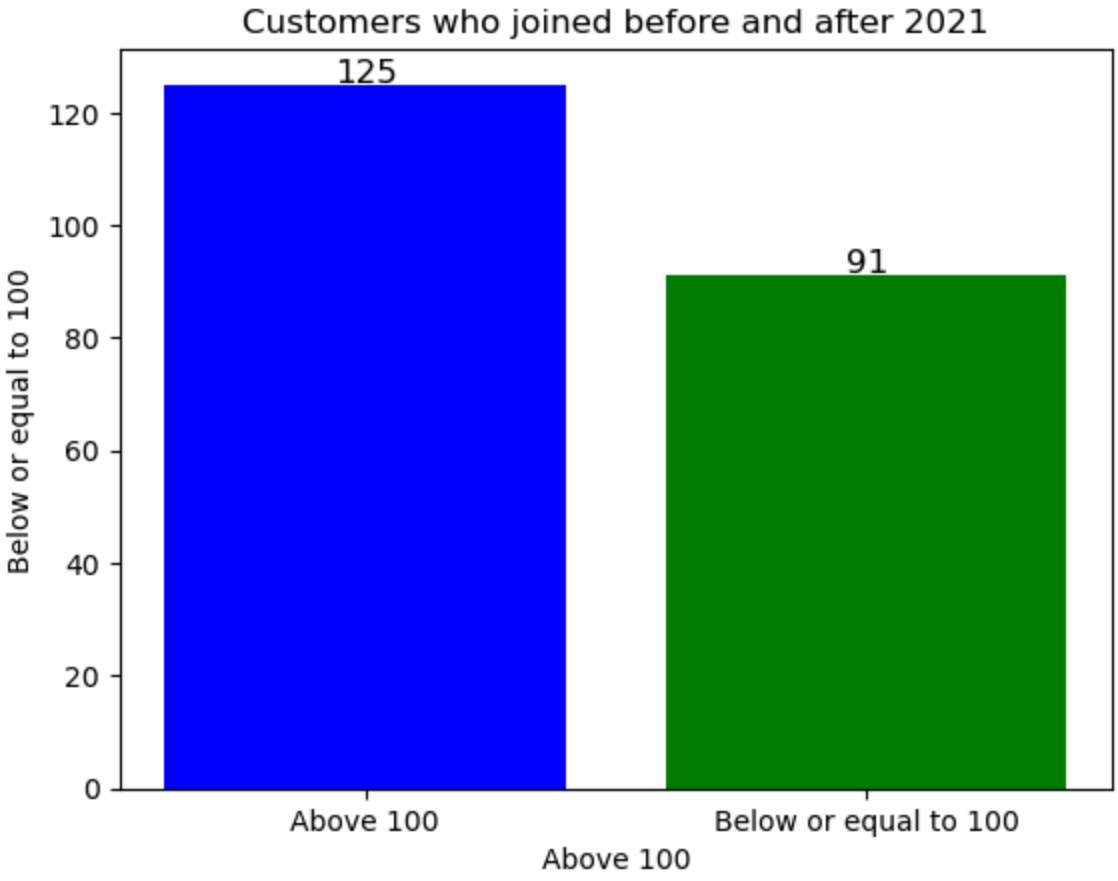
```
([<matplotlib.patches.Wedge at 0x1af3199f110>,
  <matplotlib.patches.Wedge at 0x1af3199f4d0>],
 [Text(-0.26921771464971617, 1.0665466807030923, 'Above 100'),
  Text(0.26921755360564475, -1.0665467213538242, 'Below or Equal to 100')],
 [Text(-0.14684602617257242, 0.5817527349289594, '57.9%'),
  Text(0.14684593833035167, -0.5817527571020858, '42.1%')])
```



In [122…

```
# Bar chart .. "Transactions Categories" Vs "No. of Transcations"
```

```
above_100= Transactions_df[Transactions_df['Amount']>100].shape[0]
Below_or_equal_100 = Transactions_df[Transactions_df['Amount']<=100].shape[0

# Count the number of customers in each of the category
counts={'Above 100': above_100,
       'Below or equal to 100' : Below_or_equal_100}

# create a bar chart
plt.bar(counts.keys(),counts.values(), color=['blue','green'])
plt.xlabel('Above 100')
plt.ylabel('Below or equal to 100')
plt.title('Customers who joined before and after 2021')


#Add the exact numbers on the top of the bars
for i, (key,value) in enumerate(counts.items()):
    plt.text(i,value+0.5, str(value), ha='center', fontsize=12)
plt.show()
```



In [126…

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#Reading the data
Churn_df= pd.read_csv("Churn.csv")
Churn_df.head()
```

Out[126…

	ChurnID	CustomerID	ChurnDate	Reason
0	1	22	02-01-2024	Poor Customer Service
1	2	56	06-06-2024	High Prices
2	3	45	10-10-2024	Product Quality Issues
3	4	54	12-12-2024	Lack of Product Features
4	5	82	14-02-2024	Competitor Offerings

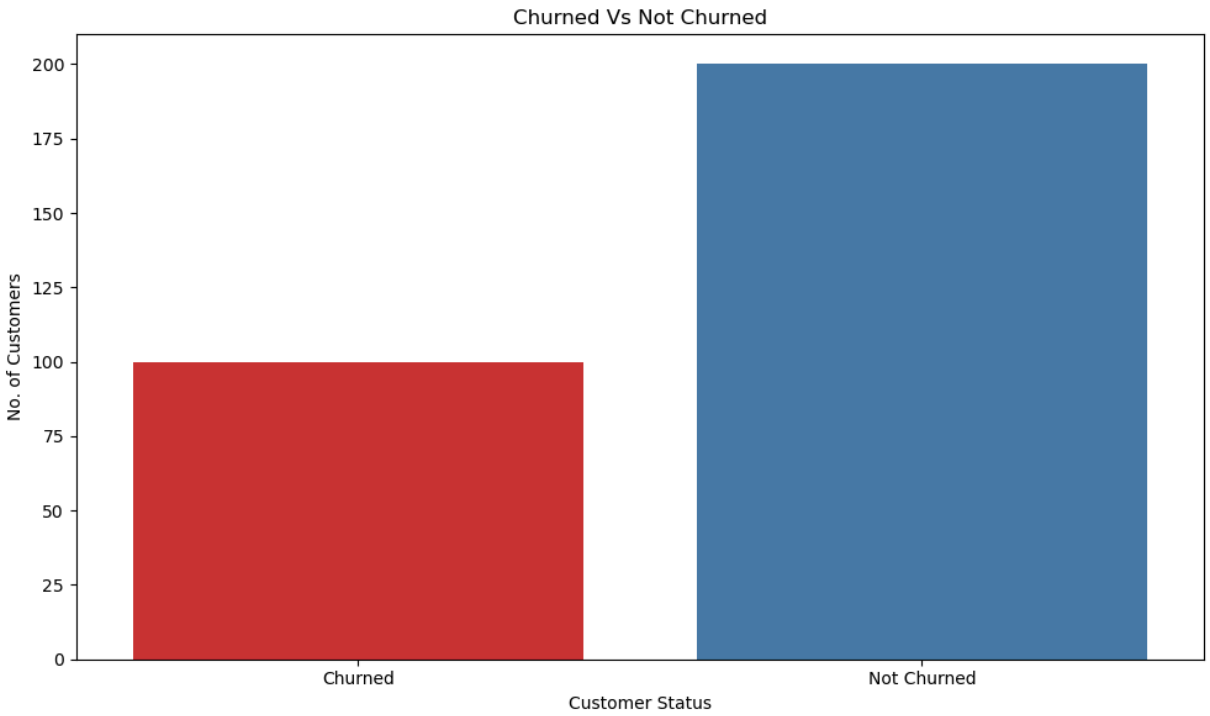
In [127…

```
#Find the number of customers who left the company/ churned for each reason
reason_counts=Churn_df['Reason'].value_counts
print(reason_counts)
```

```
<bound method IndexOpsMixin.value_counts of 0        Poor Customer Service
1                   High Prices
2       Product Quality Issues
3     Lack of Product Features
4         Competitor Offerings
                ...
95               Inconvenience
96          Lack of Engagement
97          Unmet Expectations
98             Price Increases
99      Personal Circumstances
Name: Reason, Length: 100, dtype: object>
```

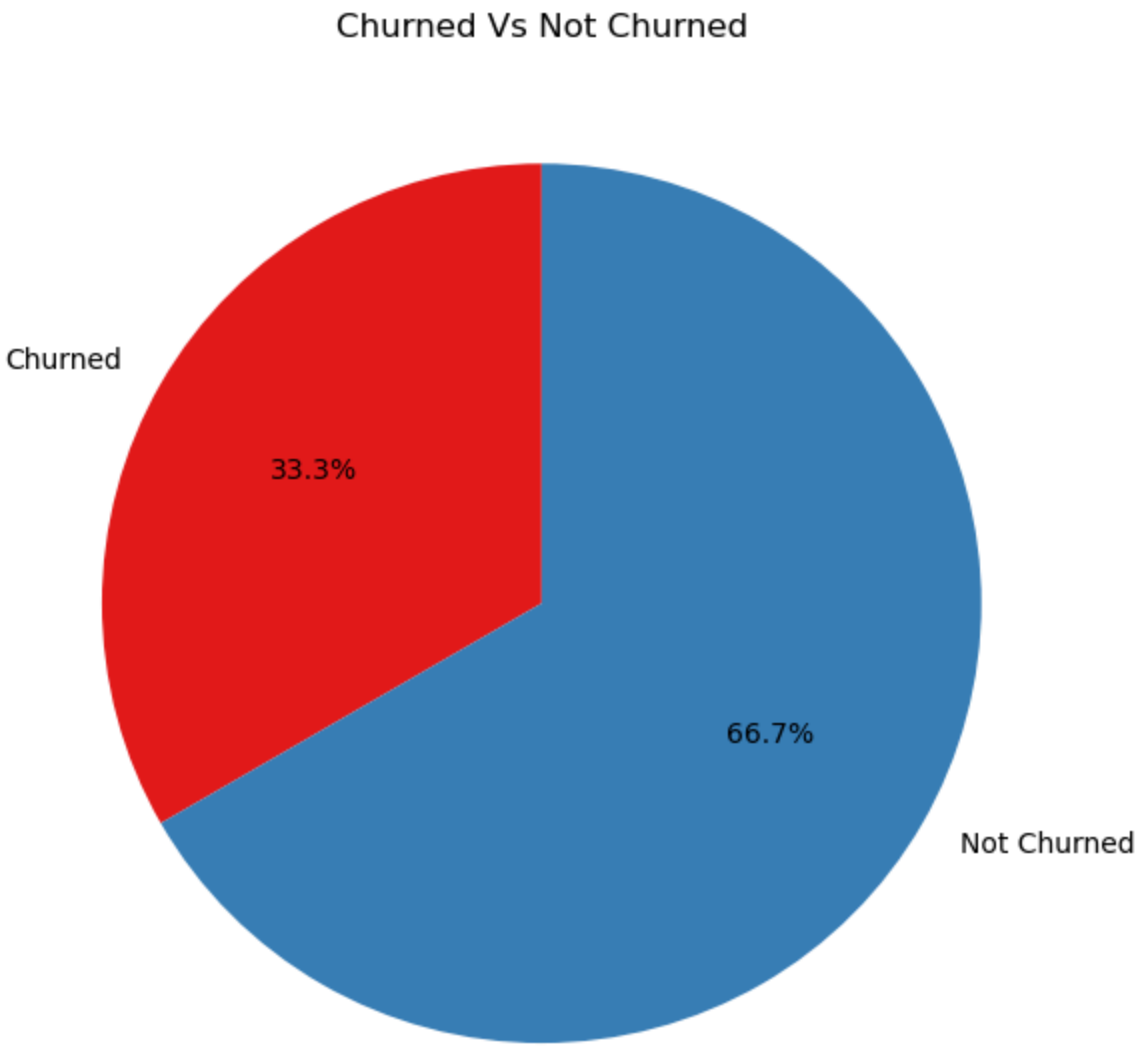
In [134…

```
# Visualization of Churned vs Not Churned customers
#Assuming we have a "Status" column indicating "Churn" and "Not Churned"

total_customers=customers_df.shape[0]
total_churned_customers=Churn_df.shape[0]
total_not_churned_customers= total_customers - total_churned_customers

#going to represent the two categories
status= pd.DataFrame({
    'Status': ['Churned', 'Not Churned'],
    'Count' : [total_churned_customers, total_not_churned_customers]
})

#Bar Chart
plt.figure(figsize=(10,6))
sns.barplot(x='Status',y='Count', data= status, palette='Set1', hue='Status'
plt.xlabel('Customer Status')
plt.ylabel('No. of Customers')
plt.title('Churned Vs Not Churned')
plt.tight_layout()
plt.show()
```



In [136…

```
plt.figure(figsize=(10,6))
plt.pie(status['Count'],labels=status['Status'],autopct='%1.1f%%',startangle
plt.title('Churned Vs Not Churned')
plt.tight_layout()
plt.show()
```



In [139…
```
#List Churn Reasons with Service"
#Filter the 'Churn_df' dataframe to include only rows where the "reason" col

service_reasons= Churn_df[Churn_df['Reason'].str.contains('service', case=Fa
print(service_reasons[['CustomerID','Reason']])
```

```
    CustomerID                Reason
0           22  Poor Customer Service
10          27  Poor Customer Service
20          72  Poor Customer Service
30         137  Poor Customer Service
40          81  Poor Customer Service
50          34  Poor Customer Service
60         102  Poor Customer Service
70          43  Poor Customer Service
80         104  Poor Customer Service
90         132  Poor Customer Service
```

In [140…
```
import pandas as pd
import numpy as np
import seaborn as sns
```

```
import matplotlib.pyplot as plt

#Reading the data
Subscriptions_df=pd.read_csv("Subscriptions.csv")
Subscriptions_df.head()
```

Out[140…

	SubscriptionID	CustomerID	StartDate	EndDate	PlanType
0	1	1	10-01-2022	09-01-2023	Annual
1	2	2	15-12-2021	14-12-2022	Monthly
2	3	3	20-03-2020	19-03-2021	Annual
3	4	4	25-06-2019	24-06-2020	Annual
4	5	5	14-07-2021	13-07-2022	Monthly

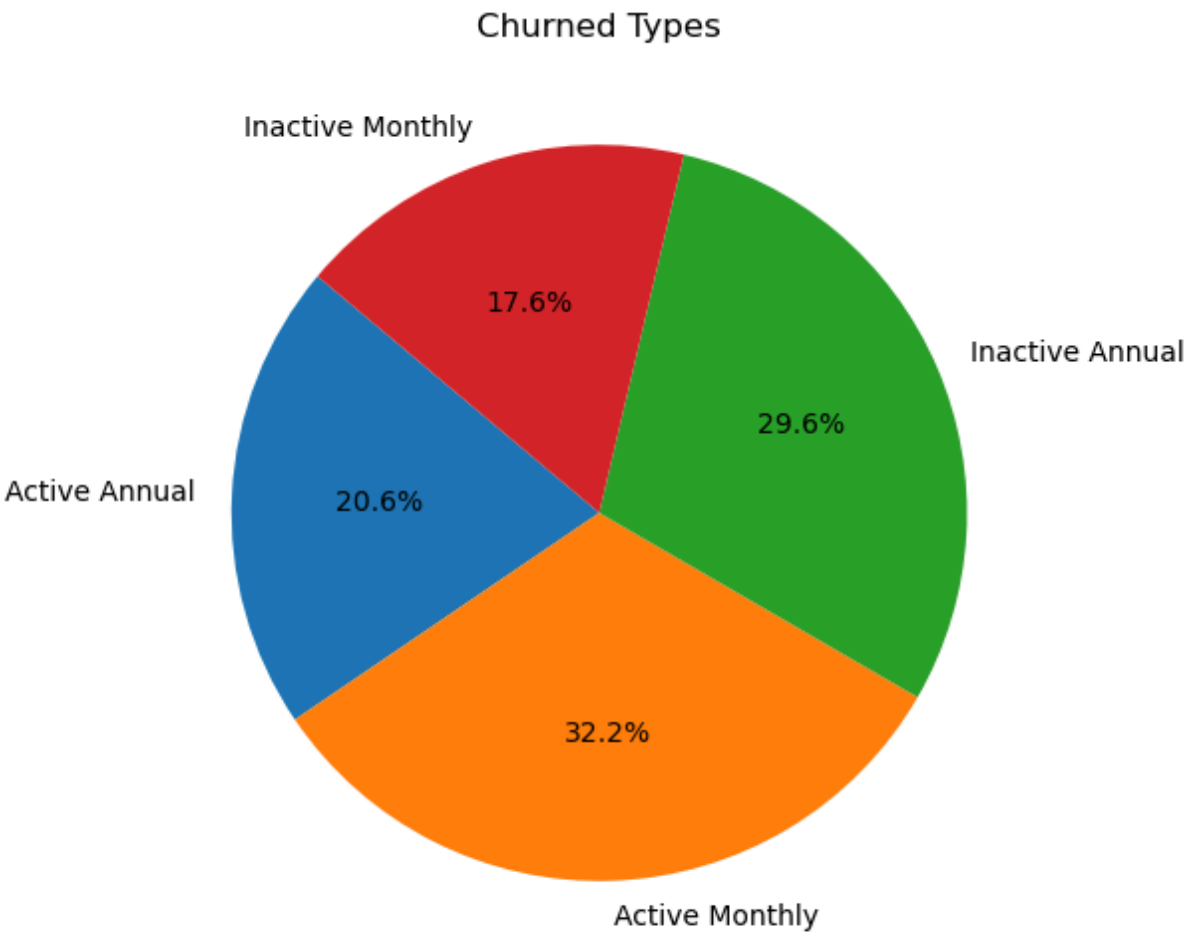
In [144…

```
# Join Customers and Subscriptions
join_df= pd.merge(customers_df, Subscriptions_df, on='CustomerID', how='inne
print(join_df)
```

```
     CustomerID FirstName LastName                    Email   PhoneNumber
\
0             1      John      Doe      john.doe@example.com  123-456-7890
1             1      John      Doe      john.doe@example.com  123-456-7890
2             1      John      Doe      john.doe@example.com  123-456-7890
3             2      Jane    Smith    jane.smith@example.com  098-765-4321
4             2      Jane    Smith    jane.smith@example.com  098-765-4321
..          ...       ...      ...                      ...           ...
394         296     Emily    Brown   emily.brown@example.com  456-789-0123
395         297      Liam   Wilson   liam.wilson@example.com  567-890-1234
396         298    Olivia    Perry  olivia.perry@example.com  678-901-2345
397         299      Ella  Johnson  ella.johnson@example.com  789-012-3456
398         300     Lucas      Lee     lucas.lee@example.com  890-123-4567

       JoinDate    Status         Region  SubscriptionID   StartDate  \
0    10-01-2022    Active  North America               1  10-01-2022
1    10-01-2022    Active  North America              44  10-01-2023
2    10-01-2022    Active  North America             101  01-07-2023
3    15-12-2021  Inactive         Europe               2  15-12-2021
4    15-12-2021  Inactive         Europe              45  15-12-2022
..          ...       ...            ...             ...         ...
394  19-03-2021  Inactive           Asia             395  01-01-2023
395  10-08-2020    Active  North America             396  01-02-2023
396  12-06-2019  Inactive         Europe             397  01-03-2023
397  18-05-2022    Active           Asia             398  01-04-2023
398  01-11-2021  Inactive  North America             399  01-05-2023

        EndDate PlanType
0    09-01-2023   Annual
1    09-01-2024  Monthly
2    30-06-2024  Monthly
3    14-12-2022  Monthly
4    14-12-2023   Annual
..          ...      ...
394  01-01-2024   Annual
395  01-02-2024  Monthly
396  01-03-2024   Annual
397  01-04-2024  Monthly
398  01-05-2024   Annual

[399 rows x 12 columns]
```

In [146…

```
#visualization for different Customer Groups [Active-Annual, Active-monthly,
result= customers_df.merge(Subscriptions_df,on='CustomerID')
result['flag'] = result['Status']+" "+result['PlanType']
data=result.groupby('flag', as_index=False).agg(total_cnt=('CustomerID','cou
plt.figure(figsize=(6,6))
plt.pie(data['total_cnt'],labels=data['flag'],autopct="%1.1f%%", startangle=
plt.title("Churned Types")
plt.show()
```



In [10]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

Transactions_df= pd.read_csv("Transactions.csv")
Transactions_df.head()
```

Out[10]:

	TransactionID	CustomerID	TransactionDate	Amount	TransactionType
0	1	15	01-01-2024	32	Purchase
1	2	22	02-01-2024	52	Refund
2	3	43	03-01-2024	165	Purchase
3	4	87	04-01-2024	134	Purchase
4	5	34	05-01-2024	158	Purchase

In [11]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
customers_df= pd.read_csv("customers.csv")
customers_df.head()
```

Out[11]:

	CustomerID	FirstName	LastName	Email	PhoneNumber
0	1	John	Doe	john.doe@example.com	123-456-7890
1	2	Jane	Smith	jane.smith@example.com	098-765-4321
2	3	Alice	Johnson	alice.j@example.com	567-890-1234
3	4	Bob	Brown	bob.brown@example.com	234-567-8901
4	5	Charlie	Davis	charlie.d@example.com	345-678-9012

In [10]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

Transactions_df= pd.read_csv("Transactions.csv")
Transactions_df.head()
```

Out[10]:

	TransactionID	CustomerID	TransactionDate	Amount	TransactionType
0	1	15	01-01-2024	32	Purchase
1	2	22	02-01-2024	52	Refund
2	3	43	03-01-2024	165	Purchase
3	4	87	04-01-2024	134	Purchase
4	5	34	05-01-2024	158	Purchase

In [12]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

Subscriptions_df= pd.read_csv("Subscriptions.csv")
Subscriptions_df.head()
```

Out[12]:

	SubscriptionID	CustomerID	StartDate	EndDate	PlanType
0	1	1	10-01-2022	09-01-2023	Annual
1	2	2	15-12-2021	14-12-2022	Monthly
2	3	3	20-03-2020	19-03-2021	Annual
3	4	4	25-06-2019	24-06-2020	Annual
4	5	5	14-07-2021	13-07-2022	Monthly

In [13]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

Churn_df= pd.read_csv("Churn.csv")
Churn_df.head()
```

Out[13]:

	ChurnID	CustomerID	ChurnDate	Reason
0	1	22	02-01-2024	Poor Customer Service
1	2	56	06-06-2024	High Prices
2	3	45	10-10-2024	Product Quality Issues
3	4	54	12-12-2024	Lack of Product Features
4	5	82	14-02-2024	Competitor Offerings

In [14]:
```
#EDA - Exploratory Data Analysis
#Data Cleaning
# Merge All Datasets

merged_df= pd.merge(customers_df, Subscriptions_df,on='CustomerID', how='lef
merged_df=pd.merge(merged_df,Transactions_df,on='CustomerID',how='left')
final_df=pd.merge(merged_df,Churn_df,on='CustomerID',how='left')
#Display the result
final_df.head()
```

Out[14]:

	CustomerID	FirstName	LastName	Email	PhoneNumber
0	1	John	Doe	john.doe@example.com	123-456-7890
1	1	John	Doe	john.doe@example.com	123-456-7890
2	1	John	Doe	john.doe@example.com	123-456-7890
3	2	Jane	Smith	jane.smith@example.com	098-765-4321
4	2	Jane	Smith	jane.smith@example.com	098-765-4321

In [15]:
```
final_df.shape
```

Out[15]: (443, 19)

In [16]:
```
#Add a column named as "churned" based on the presence of "churnID"
final_df['Churned']=final_df['ChurnID'].apply(lambda x:1 if pd.notna(x)else
final_df.head()
```

Out[16]:

	CustomerID	FirstName	LastName	Email	PhoneNumber
0	1	John	Doe	john.doe@example.com	123-456-7890
1	1	John	Doe	john.doe@example.com	123-456-7890
2	1	John	Doe	john.doe@example.com	123-456-7890
3	2	Jane	Smith	jane.smith@example.com	098-765-4321
4	2	Jane	Smith	jane.smith@example.com	098-765-4321

In [156…
```
#Delete Missing values
missing_values=final_df.isnull().sum()
print("Missing values in each column:")
print(missing_values)
```

```
Missing values in each column:
CustomerID           0
FirstName            1
LastName             1
Email                2
PhoneNumber          2
JoinDate             2
Status               1
Region               1
SubscriptionID       1
StartDate            1
EndDate              1
PlanType             1
TransactionID      141
TransactionDate    141
Amount             141
TransactionType    141
ChurnID            286
ChurnDate          286
Reason             286
Churned              0
dtype: int64
```

In [18]:
```
#Fill missing values in "FirstName" and "LastName"
final_df['FirstName']=final_df['FirstName'].fillna('xyz')
final_df['LastName']=final_df['LastName'].fillna('xyz')
```

```
print("Missing values after filling:")
print(final_df[['FirstName','LastName']].isnull().sum())
```

```
Missing values after filling:
FirstName    0
LastName     0
dtype: int64
```

In [19]:
```
#Filter rows where 'Email' or 'PhoneNumber' is missing
missing_email_phone=final_df[final_df['Email'].isnull() | final_df['PhoneNum

print("Rows with missing phone number & email:")
print(missing_email_phone)
```

```
Rows with missing phone number & email:
     CustomerID FirstName LastName                    Email   PhoneNumber
\
395         253    Robert    Jones  robert.jones@example.com           NaN
398         256     Sarah    Davis                      NaN  456-789-0123
408         266    Sophia  Johnson                      NaN  567-890-1234
410         268     Ethan    Davis   ethan.davis@example.com           NaN

       JoinDate    Status  Region  SubscriptionID   StartDate     EndDate  \
395  01-05-2021    Active    Asia           352.0  01-06-2023  01-06-2024
398  30-03-2022  Inactive  Europe           355.0  01-09-2023  01-09-2024
408  11-11-2021  Inactive    Asia           365.0  01-07-2022  01-07-2023
410  12-12-2019  Inactive  Europe           367.0  01-09-2022  01-09-2023

    PlanType  TransactionID TransactionDate  Amount TransactionType  ChurnID
\
395  Monthly          199.0      22-08-2024   114.0        Purchase      NaN
398   Annual          202.0      25-08-2024   134.0          Refund      NaN
408   Annual          212.0      04-09-2024    46.0        Purchase      NaN
410   Annual          214.0      06-09-2024   176.0        Purchase      NaN

    ChurnDate Reason  Churned
395       NaN    NaN        0
398       NaN    NaN        0
408       NaN    NaN        0
410       NaN    NaN        0
```

In [20]:
```
#Drop rows where Churned=0
final_df.dropna(subset=['Email','PhoneNumber'], how='any',inplace=True)

print(final_df[['Email','PhoneNumber']].isnull().sum())
```

```
Email          0
PhoneNumber    0
dtype: int64
```

In [21]:
```
#Handle JoinDate , Status and Region
missing_values_row=final_df[final_df[['JoinDate','Status','Region']].isnull(
print("Rows with missing JoinDate, Status & Region:")
missing_values_row
```

```
Rows with missing JoinDate, Status & Region:
```

Out[21]:

	CustomerID	FirstName	LastName	Email	PhoneNu
396	254	Emily	Taylor	emily.taylor@example.com	234-567-
397	255	Michael	Brown	michael.brown@example.com	345-678-
407	265	Hannah	Moore	hannah.moore@example.com	456-789-

In [24]:
```
#drop rows where both JoinDate, Status, and Region is missing:
final_df.dropna(subset=['JoinDate','Status','Region'], how='any',inplace=Tru

print(final_df[['JoinDate','Status','Region']].isnull().sum())
```

```
JoinDate    0
Status      0
Region      0
dtype: int64
```

In [25]:
```
final_df.shape
```

Out[25]: (436, 20)

In [26]:
```
#Handle SubscriptionID, StartDate,EndDate,and PlanType
missing_values_row=final_df[final_df[['SubscriptionID','StartDate','EndDate'
print("Rows with missing SubscriptionID, StartDate,EndDate,PlanType:")
missing_values_row
```

```
Rows with missing SubscriptionID, StartDate,EndDate,PlanType:
```

Out[26]:

	CustomerID	FirstName	LastName	Email	PhoneNumber
343	201	Ethan	Brown	ethan.b@example.com	123-456-7906

In [28]:
```
#drop rows where SubscriptionID, StartDate,EndDate,and PlanType are missing:
final_df.dropna(subset=['SubscriptionID','StartDate','EndDate','PlanType'],

print(final_df[['SubscriptionID','StartDate','EndDate','PlanType']].isnull()
```

```
SubscriptionID    0
StartDate         0
EndDate           0
PlanType          0
dtype: int64
```

In [29]:
```
final_df.shape
```

Out[29]: (435, 20)

In [31]:
```
#Set "TransactionID" to one less than the minimum existing transaction ID
#Set "TransactionDate" to a date 10 years before the minimum existing transa
#Set Amount to 0
#Set Transaction Type to 'No Transaction'
```

```
import pandas as pd
from datetime import timedelta

#Ensure TransactionDate is in datetime format,
Transactions_df['TransactionDate']=pd.to_datetime(Transactions_df['Transacti

#Find the minimum TransactionDate and TransactionDate
min_transaction_id= Transactions_df['TransactionID'].min()
min_transaction_date=Transactions_df['TransactionDate'].min()

#Define the date 10 years before the minimum transaction Date
ten_years_prior= min_transaction_date - timedelta(days=365*10)

#Fill missing values for customers with no transactions
final_df.loc[final_df['TransactionID'].isnull(),'TransactionID']=min_transac
final_df.loc[final_df['TransactionDate'].isnull(),'TransactionDate']=ten_yea
final_df.loc[final_df['Amount'].isnull(),'Amount']=0
final_df.loc[final_df['TransactionType'].isnull(),'TransactionType']='No Tra

print(final_df[['TransactionID','TransactionDate','Amount','TransactionType'
```

```
TransactionID      0
TransactionDate    0
Amount             0
TransactionType    0
dtype: int64
```

In [32]:
```
missing_values=final_df.isnull().sum()
print("Missing values in each column:")
print(missing_values)
```

```
Missing values in each column:
CustomerID           0
FirstName            0
LastName             0
Email                0
PhoneNumber          0
JoinDate             0
Status               0
Region               0
SubscriptionID       0
StartDate            0
EndDate              0
PlanType             0
TransactionID        0
TransactionDate      0
Amount               0
TransactionType      0
ChurnID            278
ChurnDate          278
Reason             278
Churned              0
dtype: int64
```

In [33]:
```
final_df.shape
```

Out[33]: (435, 20)

In [38]:
```
import pandas as pd
from datetime import timedelta

#Ensure TransactionDate is in datetime format,
Churn_df['ChurnDate']=pd.to_datetime(Churn_df['ChurnDate'],errors='coerce')

#Find the minimum TransactionDate and TransactionDate
min_churn_id= Churn_df['ChurnID'].min()
min_churn_date=Churn_df['ChurnDate'].min()

#Define the date 10 years before the minimum transaction Date
ten_years_prior= min_churn_date - timedelta(days=365*10)

#Fill missing values for customers with no transactions
final_df.loc[final_df['ChurnID'].isnull(),'ChurnID']=min_churn_id-1
final_df.loc[final_df['ChurnDate'].isnull(),'ChurnDate']=ten_years_prior.str
final_df.loc[final_df['Reason'].isnull(),'Reason']='Unknown Reason'

print(final_df[['ChurnID','ChurnDate','Reason']].isnull().sum())
```

```
ChurnID      0
ChurnDate    0
Reason       0
dtype: int64
```

In [39]:
```
missing_values=final_df.isnull().sum()
print("Missing values in each column:")
print(missing_values)
```

```
Missing values in each column:
CustomerID         0
FirstName          0
LastName           0
Email              0
PhoneNumber        0
JoinDate           0
Status             0
Region             0
SubscriptionID     0
StartDate          0
EndDate            0
PlanType           0
TransactionID      0
TransactionDate    0
Amount             0
TransactionType    0
ChurnID            0
ChurnDate          0
Reason             0
Churned            0
dtype: int64
```

In [40]:
```
final_df.shape
```

Out[40]: (435, 20)

In [ ]: