In [4]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import SMOTE
from tpot import TPOTClassifier
import shap
```

In [8]:
```python
df = pd.read_csv('C:\\Users\\Admin\\OneDrive\\Desktop\\Customertravel.csv')
print(df)
```

```
     Age FrequentFlyer AnnualIncomeClass  ServicesOpted  \
0     34            No     Middle Income              6
1     34           Yes        Low Income              5
2     37            No     Middle Income              3
3     30            No     Middle Income              2
4     30            No        Low Income              1
..   ...           ...               ...            ...
949   31           Yes        Low Income              1
950   30            No     Middle Income              5
951   37            No     Middle Income              4
952   30            No        Low Income              1
953   31           Yes       High Income              1

    AccountSyncedToSocialMedia BookedHotelOrNot  Target
0                           No              Yes       0
1                          Yes               No       1
2                          Yes               No       0
3                           No               No       0
4                           No               No       0
..                         ...              ...     ...
949                         No               No       0
950                         No              Yes       0
951                         No               No       0
952                        Yes              Yes       0
953                         No               No       0

[954 rows x 7 columns]
```

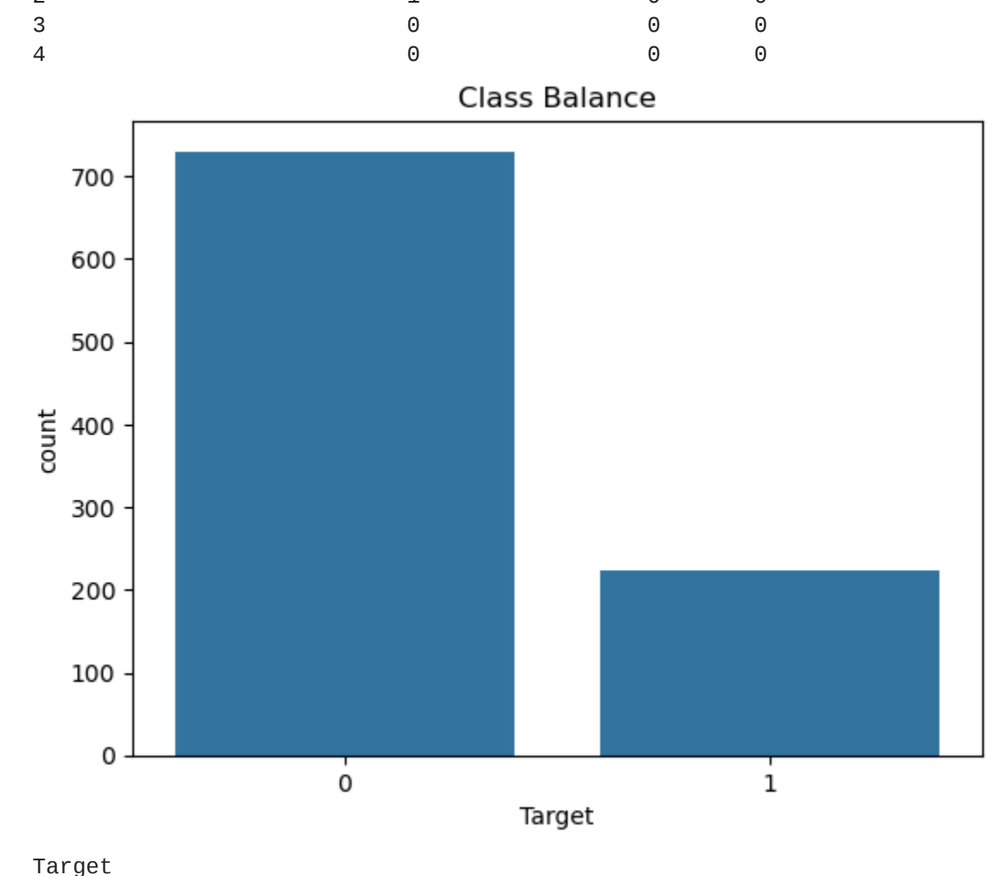
In [10]:
```python
print(df.isnull().sum())

# Step 1.1: Handle missing values for numeric columns only
df.fillna(df.mean(numeric_only=True), inplace=True)

# Check for missing values after filling
print(df.isnull().sum())
```

```
Age                           0
FrequentFlyer                 0
AnnualIncomeClass             0
ServicesOpted                 0
AccountSyncedToSocialMedia    0
BookedHotelOrNot              0
Target                        0
dtype: int64
Age                           0
FrequentFlyer                 0
AnnualIncomeClass             0
ServicesOpted                 0
AccountSyncedToSocialMedia    0
BookedHotelOrNot              0
Target                        0
dtype: int64
```

In [12]:
```python
label_encoder = LabelEncoder()
categorical_cols = ['FrequentFlyer', 'AnnualIncomeClass', 'AccountSyncedToSocialMedia', 'BookedHotelOrNot']
for col in categorical_cols:
    df[col] = label_encoder.fit_transform(df[col])
```

In [14]:
```python
scaler = StandardScaler()
df[['Age', 'ServicesOpted']] = scaler.fit_transform(df[['Age', 'ServicesOpted']])

# Display the preprocessed dataset
print(df.head())

# Check the balance of the target variable
sns.countplot(x='Target', data=df)
plt.title('Class Balance')
plt.show()

# Display the count of each class
print(df['Target'].value_counts())

# Split the data into features and target
X = df.drop('Target', axis=1)
y = df['Target']
```

```
        Age  FrequentFlyer  AnnualIncomeClass  ServicesOpted  \
0  0.566904              0                  2       2.219330
1  0.566904              2                  1       1.596429
2  1.466282              0                  2       0.350627
3 -0.632267              0                  2      -0.272274
4 -0.632267              0                  1      -0.895175

   AccountSyncedToSocialMedia  BookedHotelOrNot  Target
0                           0                 1       0
1                           1                 0       1
2                           1                 0       0
3                           0                 0       0
4                           0                 0       0
```




```
Target
0    730
1    224
Name: count, dtype: int64
```

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.preprocessing import StandardScaler, LabelEncoder from sklearn.model_selection import train_test_split from imblearn.over_sampling import SMOTE

Load the new dataset

df = pd.read_csv('C:\Users\Admin\OneDrive\Desktop\Customertravel.csv')

Handle missing values for numeric columns only

df.fillna(df.mean(numeric_only=True), inplace=True)

Encode categorical variables

label_encoder = LabelEncoder() categorical_cols = ['FrequentFlyer', 'AnnualIncomeClass', 'AccountSyncedToSocialMedia', 'BookedHotelOrNot'] for col in categorical_cols: df[col] = label_encoder.fit_transform(df[col])

Feature scaling

scaler = StandardScaler() df[['Age', 'ServicesOpted']] = scaler.fit_transform(df[['Age', 'ServicesOpted']])

Split the data into features and target

X = df.drop('Target', axis=1) y = df['Target']

Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Apply SMOTE to balance the classes in the training set

smote = SMOTE(random_state=42) X_res, y_res = smote.fit_resample(X_train, y_train)

Plot class balance before and after SMOTE

fig, ax = plt.subplots(1, 2, figsize=(12, 6))

Before SMOTE

sns.countplot(x=y_train, ax=ax[0]) ax[0].set_title('Class Balance Before SMOTE') ax[0].set_xlabel('Class') ax[0].set_ylabel('Count')

After SMOTE

sns.countplot(x=y_res, ax=ax[1]) ax[1].set_title('Class Balance After SMOTE') ax[1].set_xlabel('Class') ax[1].set_ylabel('Count')

plt.tight_layout() plt.show()

In [18]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply SMOTE to balance the classes in the training set
smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X_train, y_train)

# Check the new balance
print(pd.Series(y_res).value_counts())
```

```
Target
0    577
1    577
Name: count, dtype: int64
```

In [24]:
```python
print(df.columns)
```

```
Index(['Age', 'FrequentFlyer', 'AnnualIncomeClass', 'ServicesOpted',
       'AccountSyncedToSocialMedia', 'BookedHotelOrNot', 'Target'],
      dtype='object')
```

In [26]:
```python
from sklearn.decomposition import PCA
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Print column names to check if 'Cluster' exists
print("Columns in DataFrame:", df.columns)

# Check if 'Cluster' is in the DataFrame
if 'Cluster' in df.columns:
    pca = PCA(n_components=2)
    df_pca = pca.fit_transform(df.drop('Cluster', axis=1))

    # Create a DataFrame with PCA results
    df_pca = pd.DataFrame(data=df_pca, columns=['PCA1', 'PCA2'])
    df_pca['Cluster'] = df['Cluster']

    # Visualize the clusters
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', data=df_pca, palette='viridis', s=100)
    plt.title('Customer Segmentation using K-means Clustering')
    plt.xlabel('PCA1')
    plt.ylabel('PCA2')
    plt.legend(title='Cluster')
    plt.grid(True)
    plt.savefig('pca_clusters.png', dpi=300, bbox_inches='tight')
    plt.show()
else:
    print("'Cluster' column not found in DataFrame.")
```

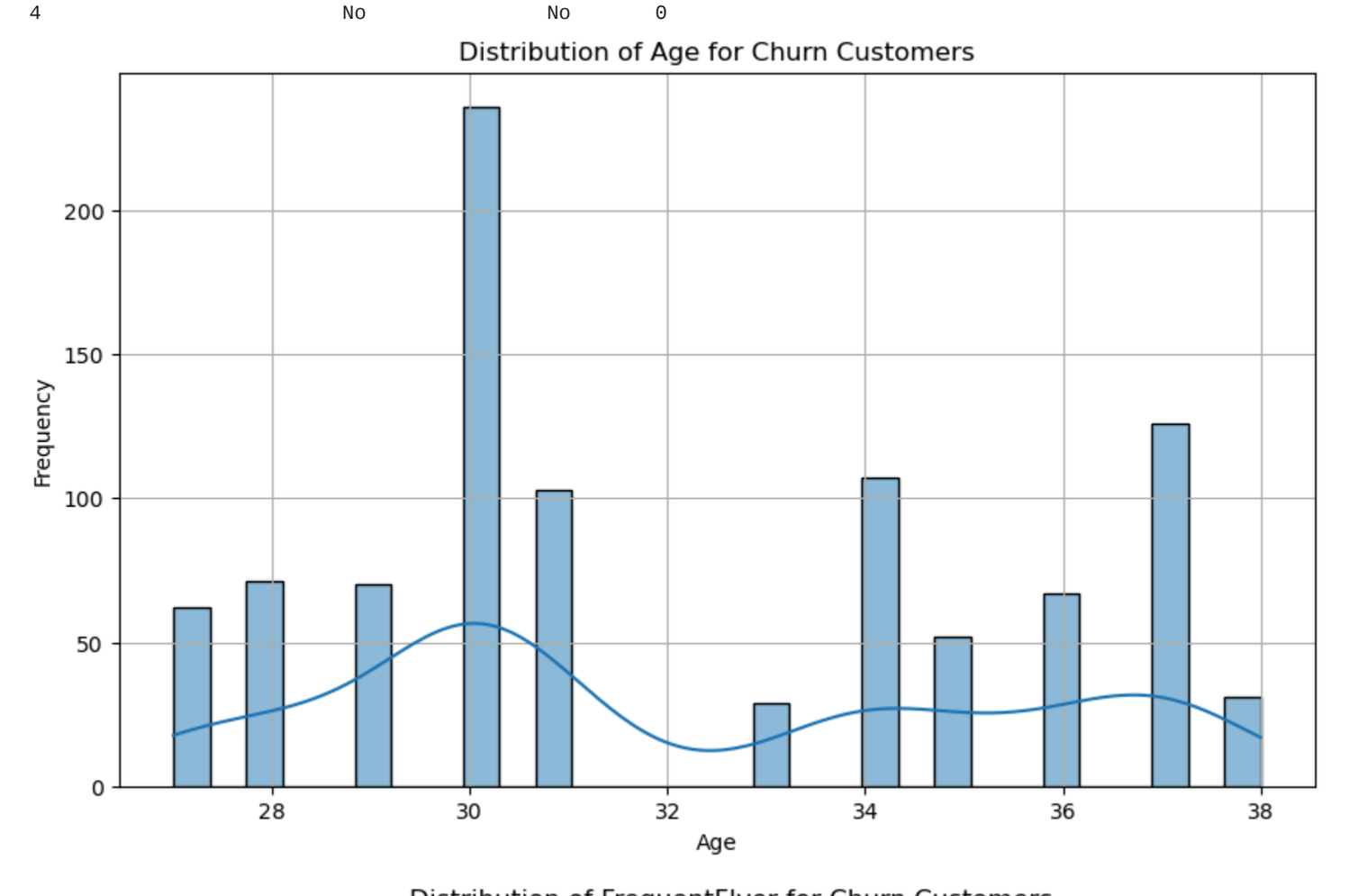
```
Columns in DataFrame: Index(['Age', 'FrequentFlyer', 'AnnualIncomeClass', 'ServicesOpted',
       'AccountSyncedToSocialMedia', 'BookedHotelOrNot', 'Target'],
      dtype='object')
'Cluster' column not found in DataFrame.
```

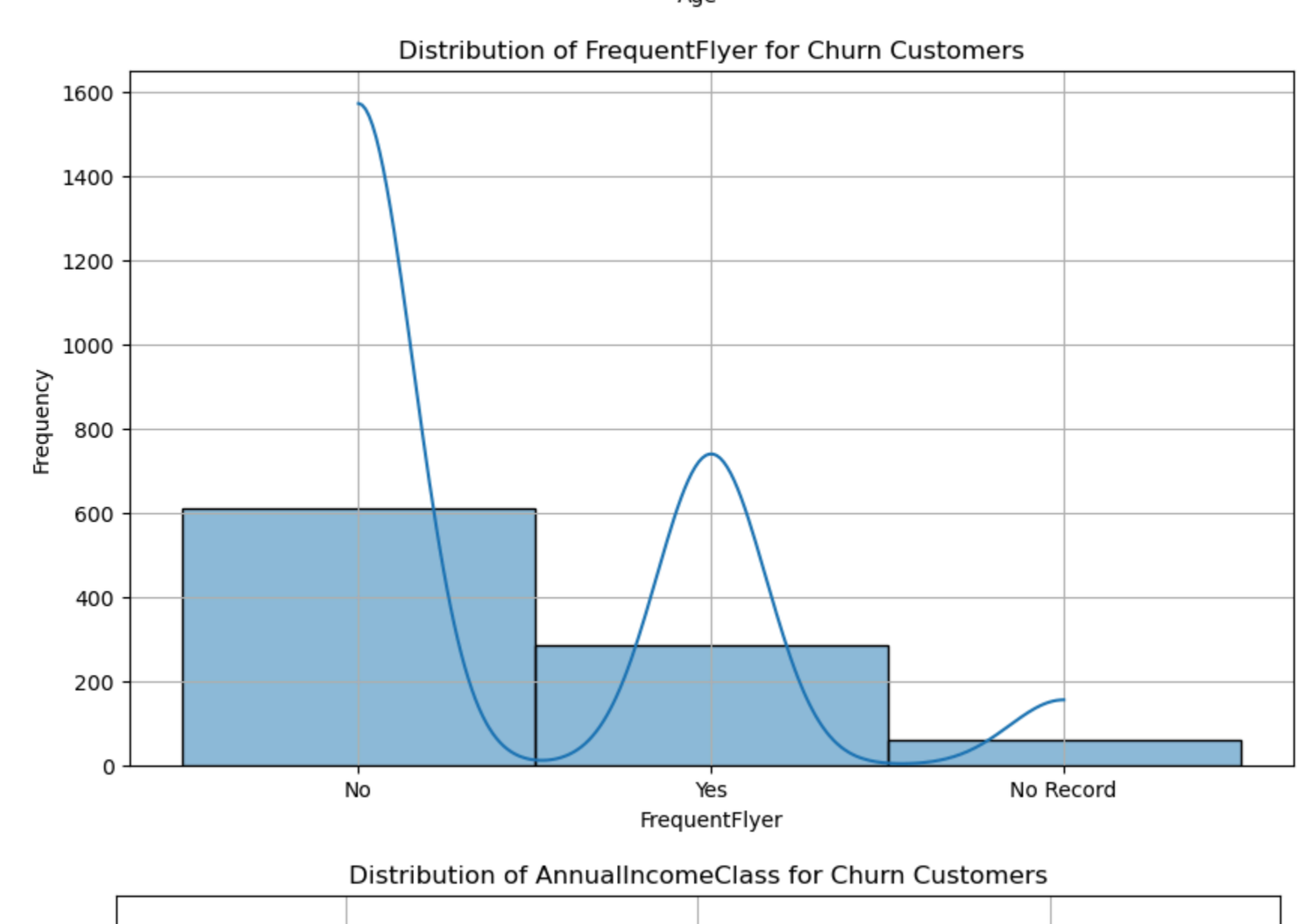
In [28]:
```python
print(df.head())
```

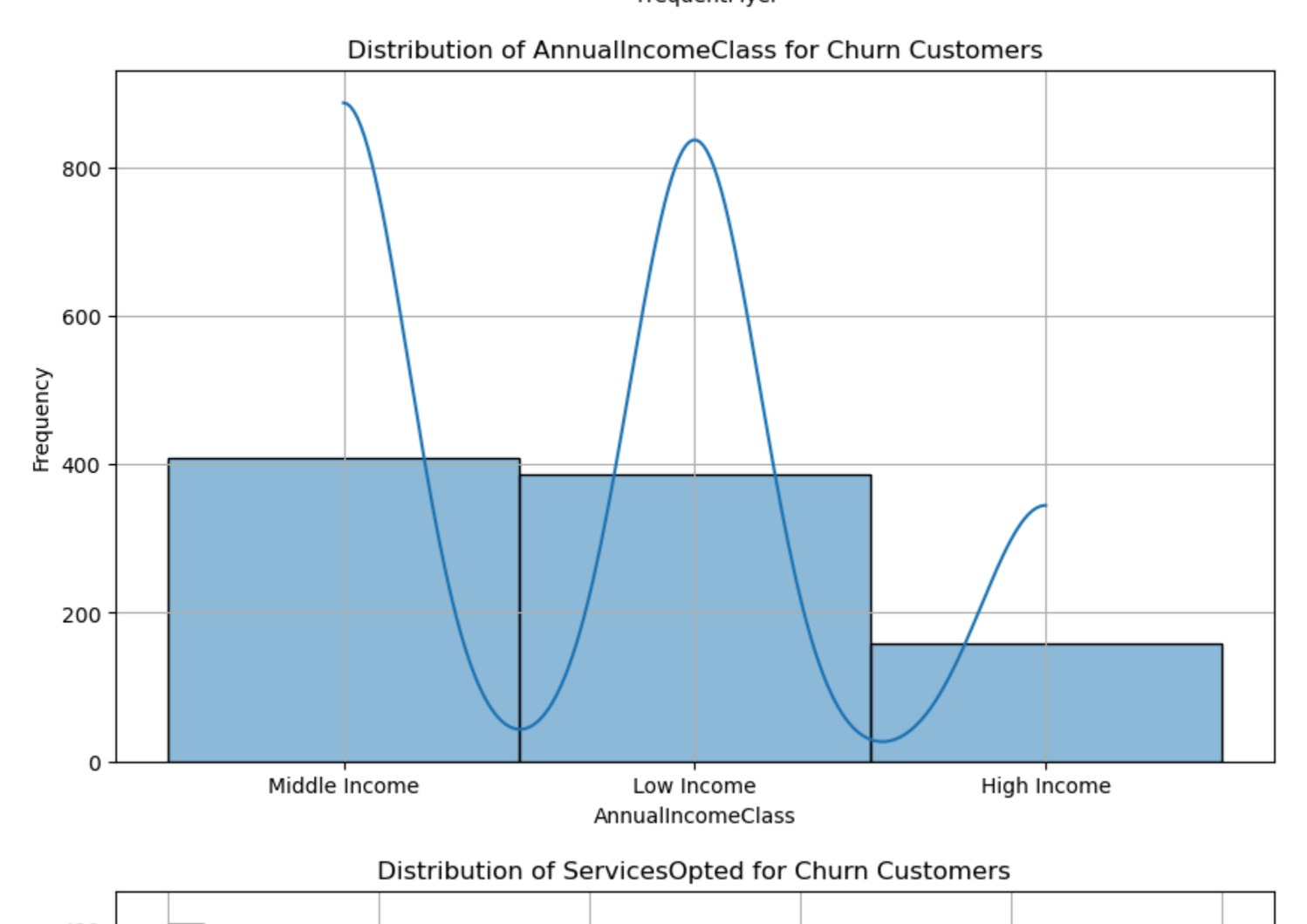
```
        Age  FrequentFlyer  AnnualIncomeClass  ServicesOpted  \
0  0.566904              0                  2       2.219330
1  0.566904              2                  1       1.596429
2  1.466282              0                  2       0.350627
3 -0.632267              0                  2      -0.272274
4 -0.632267              0                  1      -0.895175

   AccountSyncedToSocialMedia  BookedHotelOrNot  Target
0                           0                 1       0
1                           1                 0       1
2                           1                 0       0
3                           0                 0       0
4                           0                 0       0
```

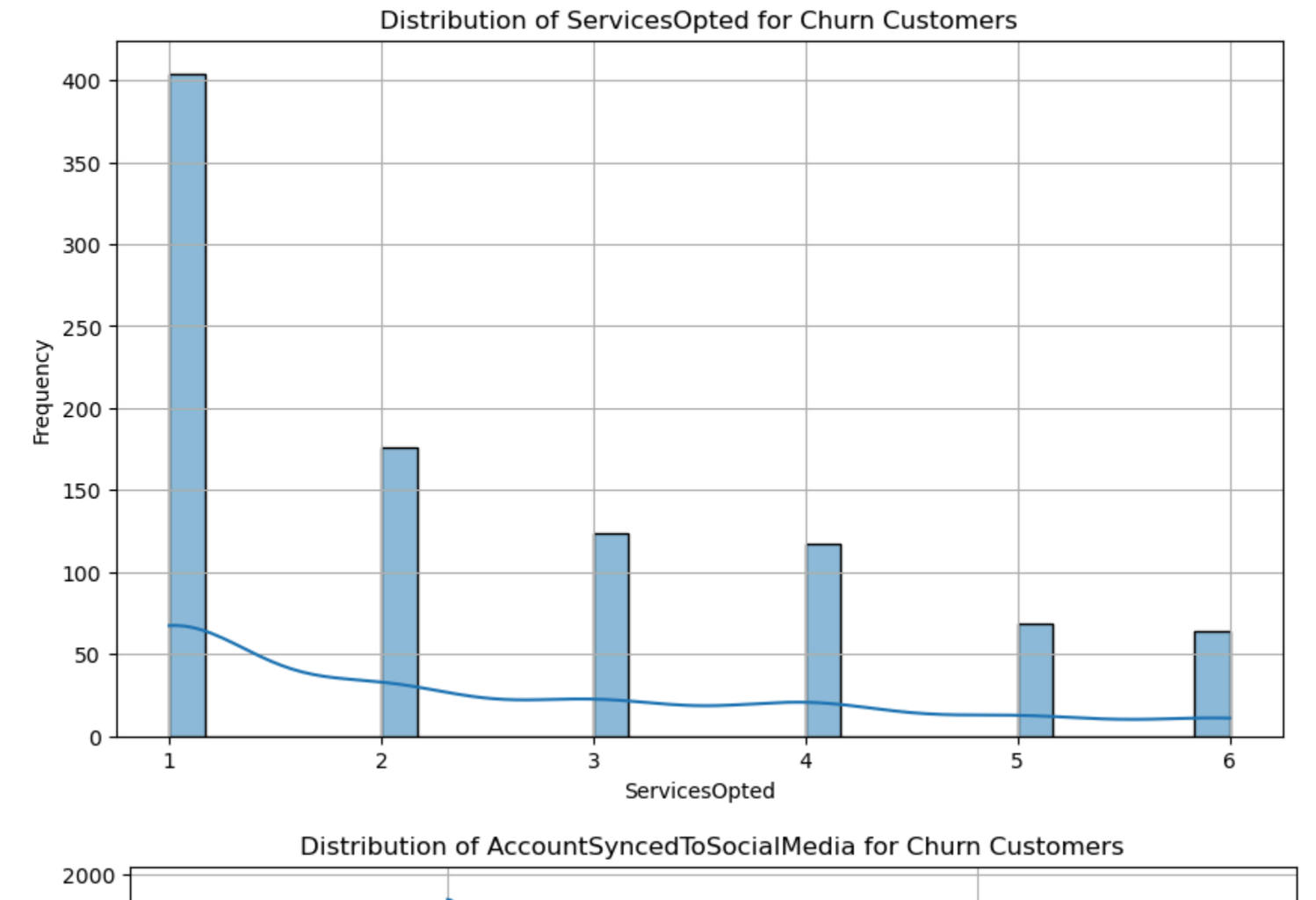
In [43]:
```python
import pandas as pd

# Example: Loading a DataFrame from a CSV file
churn_customers = pd.read_csv('C:\\Users\\Admin\\OneDrive\\Desktop\\Customertravel.csv')

# Verify DataFrame is loaded
print(churn_customers.head())
# List of important features
important_features = [
    'Age',
    'FrequentFlyer',
    'AnnualIncomeClass',
    'ServicesOpted',
    'AccountSyncedToSocialMedia',
    'BookedHotelOrNot'
]

# Visualize important features
for feature in important_features:
    if feature in churn_customers.columns:
        plt.figure(figsize=(10, 6))
        sns.histplot(churn_customers[feature], kde=True, bins=30)
        plt.title(f'Distribution of {feature} for Churn Customers')
        plt.xlabel(feature)
        plt.ylabel('Frequency')
        plt.grid(True)
        plt.show()
    else:
        print(f"Feature '{feature}' not found in DataFrame.")
```

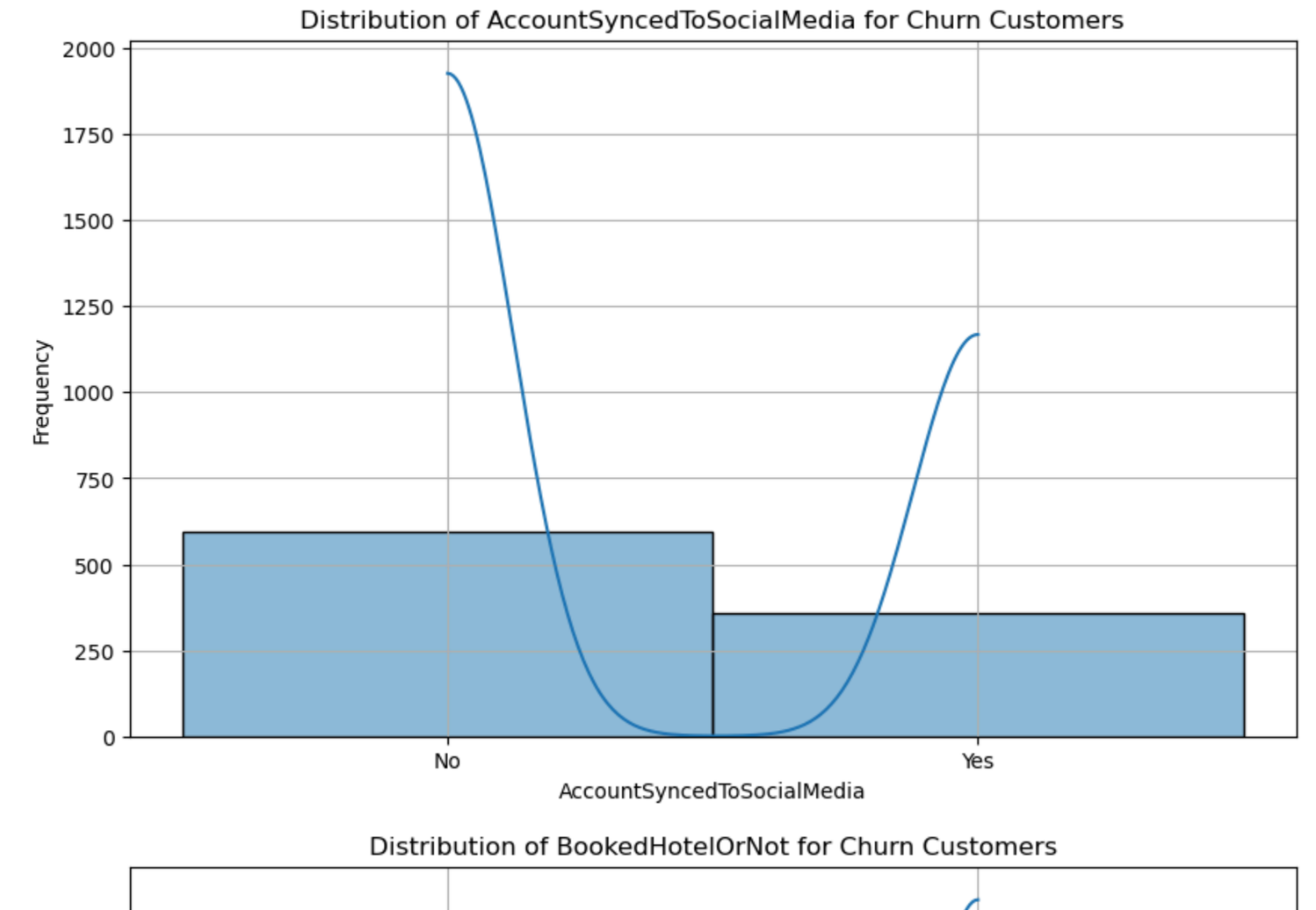
```
   Age FrequentFlyer AnnualIncomeClass  ServicesOpted  \
0   34            No     Middle Income              6
1   34           Yes        Low Income              5
2   37            No     Middle Income              3
3   30            No     Middle Income              2
4   30            No        Low Income              1

  AccountSyncedToSocialMedia BookedHotelOrNot  Target
0                         No              Yes       0
1                        Yes               No       1
2                        Yes               No       0
3                         No               No       0
4                         No               No       0
```

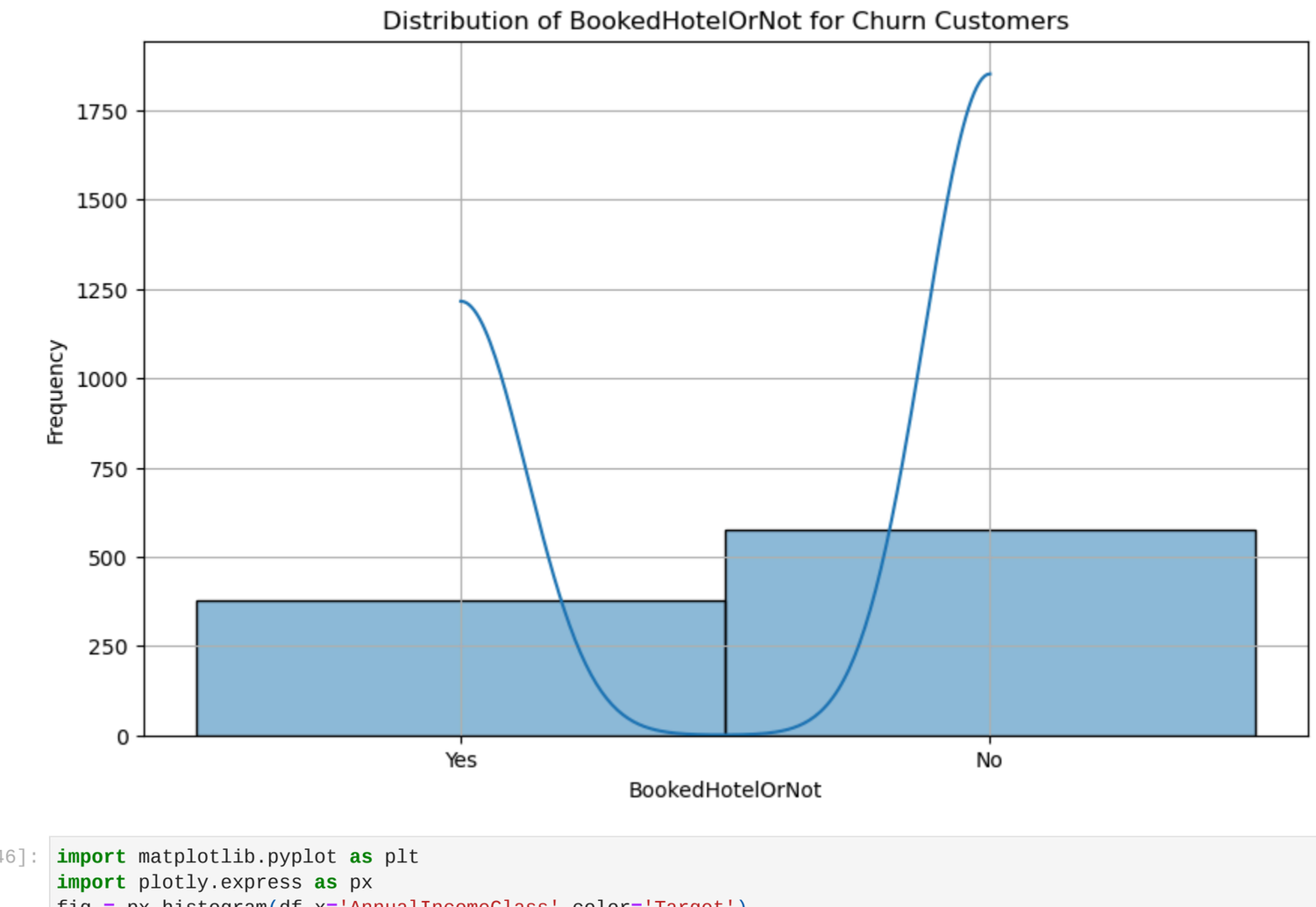




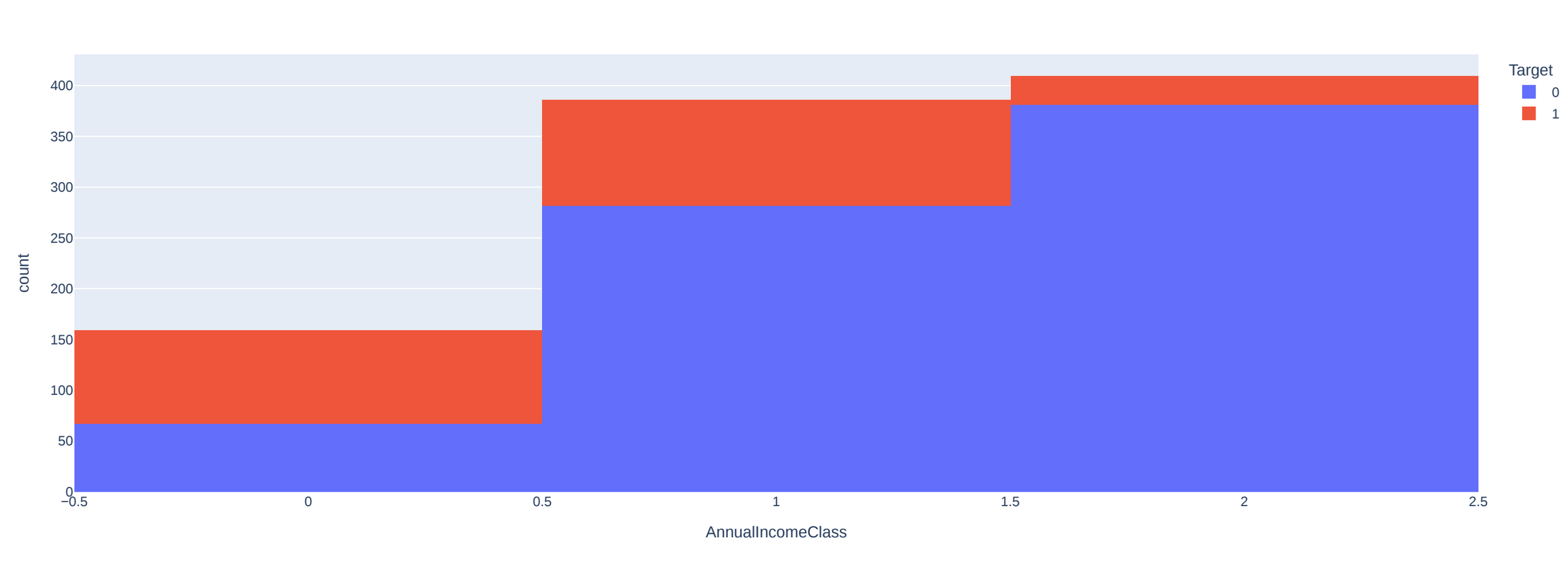




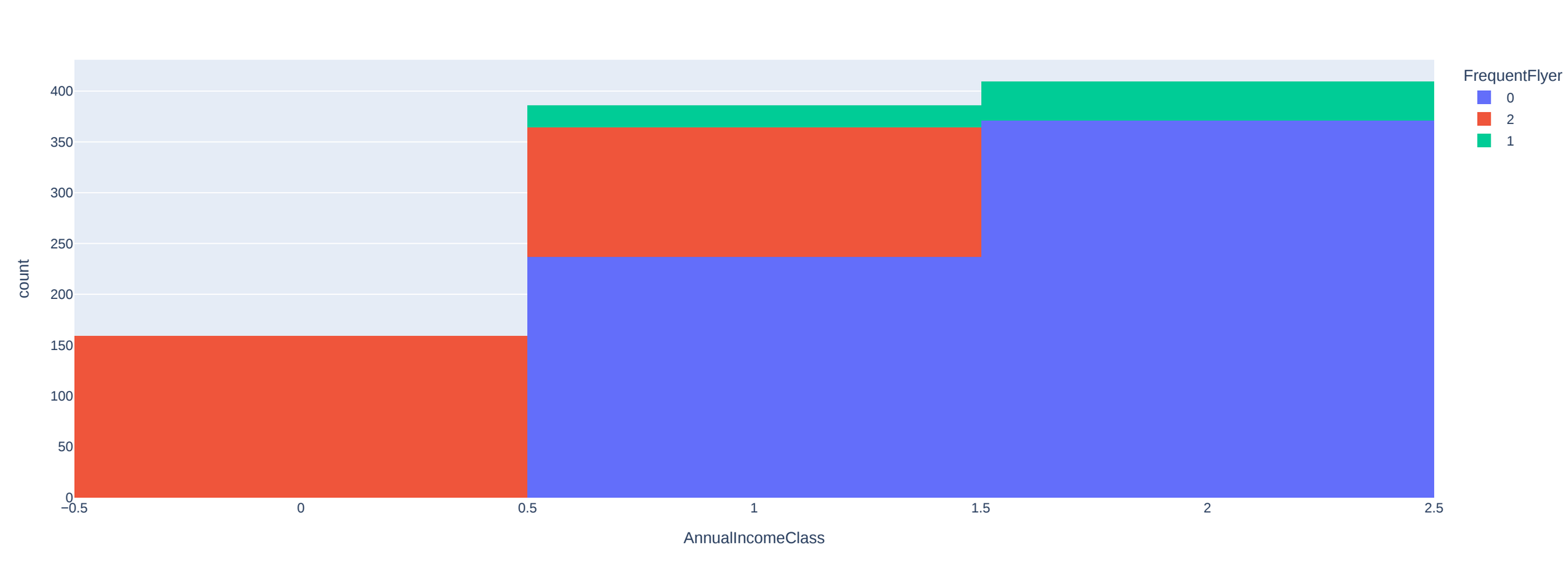




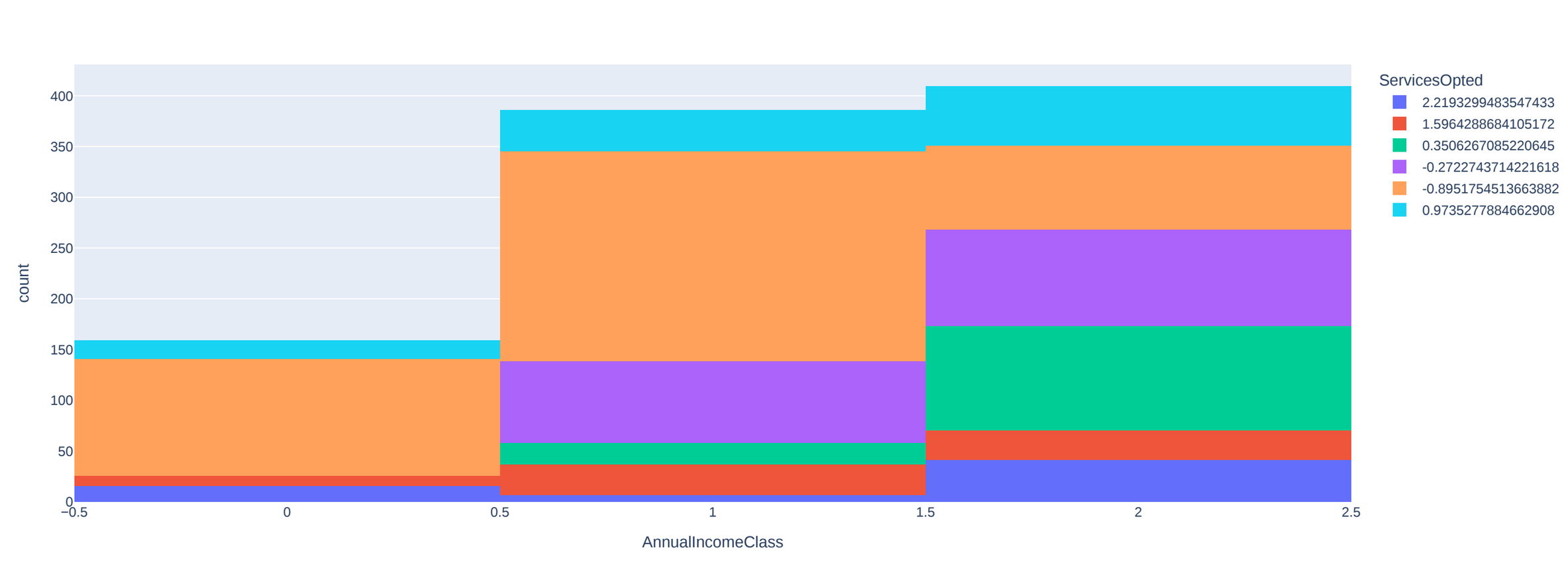
In [46]:
```python
import matplotlib.pyplot as plt
import plotly.express as px
fig = px.histogram(df,x='AnnualIncomeClass',color='Target')
fig.show()
```



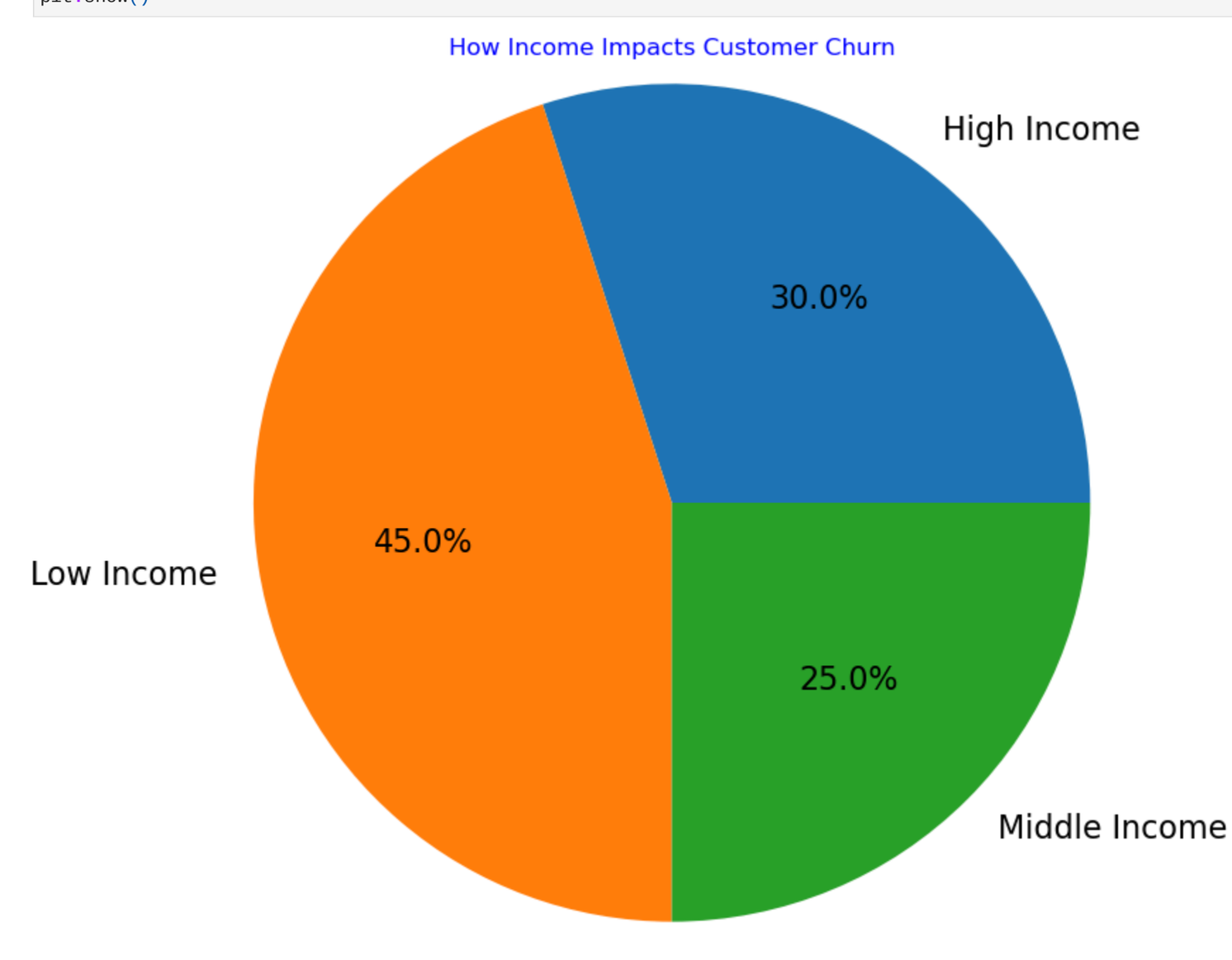
In [48]:
```python
fig = px.histogram(df,x='AnnualIncomeClass',color='FrequentFlyer')
fig.show()
```



In [50]:
```python
fig = px.histogram(df,x='AnnualIncomeClass',color='ServicesOpted')
fig.show()
```

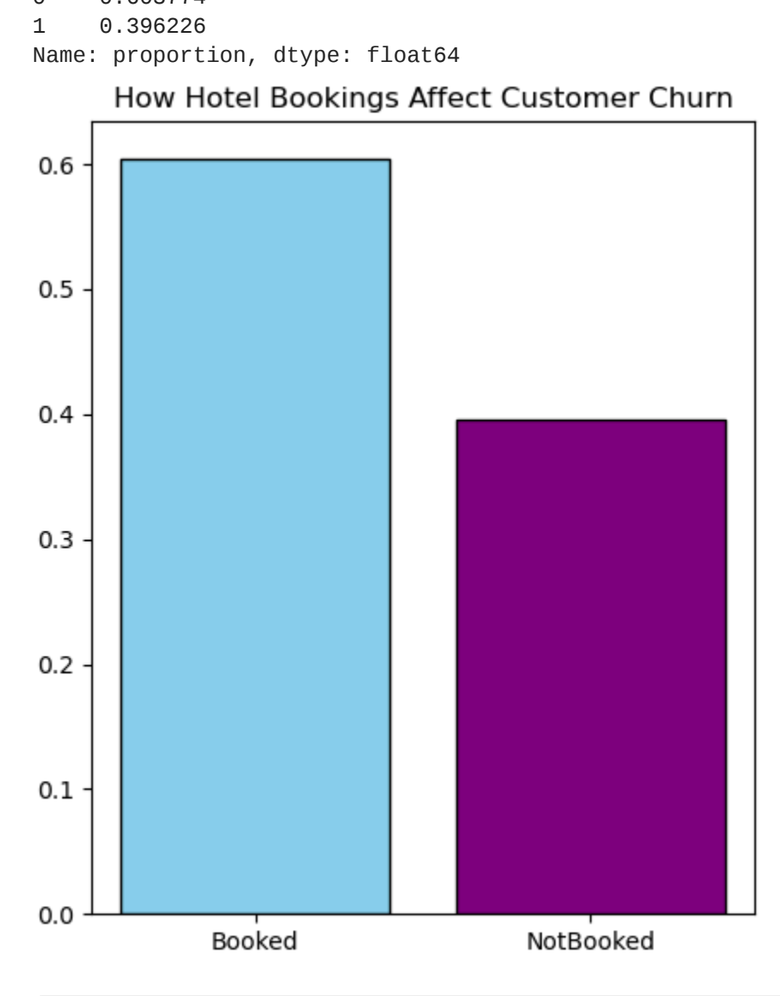


In [54]:
```python
import matplotlib.pyplot as plt

# Example data for the pie chart
# Replace this with your actual data
income = pd.Series([30, 45, 25], index=["High Income", "Low Income", "Middle Income"])

# Create the pie chart
plt.figure(figsize=(8, 8))  # Adjust the figure size as needed
plt.pie(
    income.values,
    labels=income.index,
    autopct='%1.1f%%',
    radius=1.2,
    textprops={"fontsize": 16}
)
plt.title("How Income Impacts Customer Churn", color="b")
plt.show()
```



In [65]:
```python
churn_perc = df ['BookedHotelOrNot'].value_counts(normalize = 1)
print(churn_perc)

plt.figure(figsize=(5,6))
plt.bar(["Booked","NotBooked"], df ['BookedHotelOrNot'].value_counts(normalize = 1), edgecolor = 'black', width = 0.8, color = ['skyblue', 'purple'])
plt.title('How Hotel Bookings Affect Customer Churn')
plt.show()
```

```
BookedHotelOrNot
0    0.603774
1    0.396226
Name: proportion, dtype: float64
```



In [67]:
```python
print("Statistical Summary")
df.describe().T
```

Statistical Summary

Out[67]:

	count	mean	std	min	25%	50%	75%	max
Age	954.0	-9.468317e-16	1.000525	-1.531645	-0.632267	-0.332475	0.866696	1.766075
FrequentFlyer	954.0	6.624738e-01	0.907770	0.000000	0.000000	0.000000	2.000000	2.000000
AnnualIncomeClass	954.0	1.262055e+00	0.726132	0.000000	1.000000	1.000000	2.000000	2.000000
ServicesOpted	954.0	-3.351617e-17	1.000525	-0.895175	-0.895175	-0.272274	0.973528	2.219330
AccountSyncedToSocialMedia	954.0	3.773585e-01	0.484980	0.000000	0.000000	0.000000	1.000000	1.000000
BookedHotelOrNot	954.0	3.962264e-01	0.489369	0.000000	0.000000	0.000000	1.000000	1.000000
Target	954.0	2.348008e-01	0.424097	0.000000	0.000000	0.000000	0.000000	1.000000

In [72]:
```python
print(df)
```

```
          Age  FrequentFlyer  AnnualIncomeClass  ServicesOpted  \
0    0.566904              0                  2       2.219330
1    0.566904              2                  1       1.596429
2    1.466282              0                  2       0.350627
3   -0.632267              0                  2      -0.272274
4   -0.632267              0                  1      -0.895175
..        ...            ...                ...            ...
949 -0.332475              2                  1      -0.895175
950 -0.632267              0                  2       1.596429
951  1.466282              0                  2       0.973528
952 -0.632267              0                  1      -0.895175
953 -0.332475              2                  0      -0.895175

     AccountSyncedToSocialMedia  BookedHotelOrNot  Target
0                             0                 1       0
1                             1                 0       1
2                             1                 0       0
3                             0                 0       0
4                             0                 0       0
..                          ...               ...     ...
949                           0                 0       0
950                           0                 1       0
951                           0                 0       0
952                           1                 1       0
953                           0                 0       0

[954 rows x 7 columns]
```

In [74]:
```python
print(churn_perc)
```

```
BookedHotelOrNot
0    0.603774
1    0.396226
Name: proportion, dtype: float64
```