In [15]:
```
# Importing dependencies

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [16]:
```
# Load the dataset

df = pd.read_csv("C:/Users/HP/Downloads/Netflix Userbase.csv")
```

In [17]:
```
df
```

Out[17]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Plan Duration
0	1	Basic	10	15-01-22	10-06-23	United States	28	Male	Smartphone	1 Month
1	2	Premium	15	05-09-21	22-06-23	Canada	35	Female	Tablet	1 Month
2	3	Standard	12	28-02-23	27-06-23	United Kingdom	42	Male	Smart TV	1 Month
3	4	Standard	12	10-07-22	26-06-23	Australia	51	Female	Laptop	1 Month
4	5	Basic	10	01-05-23	28-06-23	Germany	33	Male	Smartphone	1 Month
...	...	...	...	...	...	...	...	...	...	...
2495	2496	Premium	14	25-07-22	12-07-23	Spain	28	Female	Smart TV	1 Month
2496	2497	Basic	15	04-08-22	14-07-23	Spain	33	Female	Smart TV	1 Month
2497	2498	Standard	12	09-08-22	15-07-23	United States	38	Male	Laptop	1 Month
2498	2499	Standard	13	12-08-22	12-07-23	Canada	48	Female	Tablet	1 Month
2499	2500	Basic	15	13-08-22	12-07-23	United States	35	Female	Smart TV	1 Month

2500 rows × 10 columns

In [18]:
```
df.shape
```

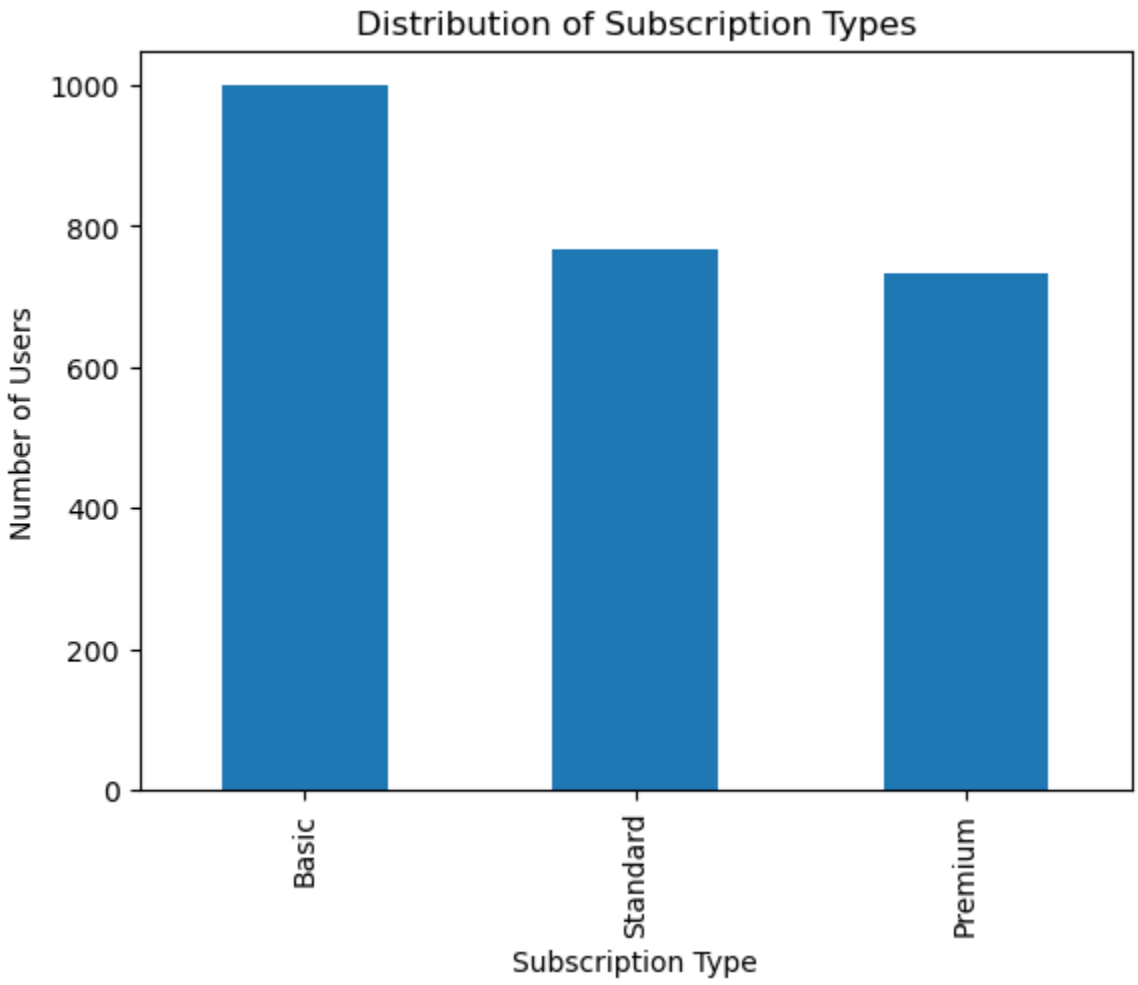
Out[18]:
```
(2500, 10)
```

In [19]:
```
# Data Cleaning

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

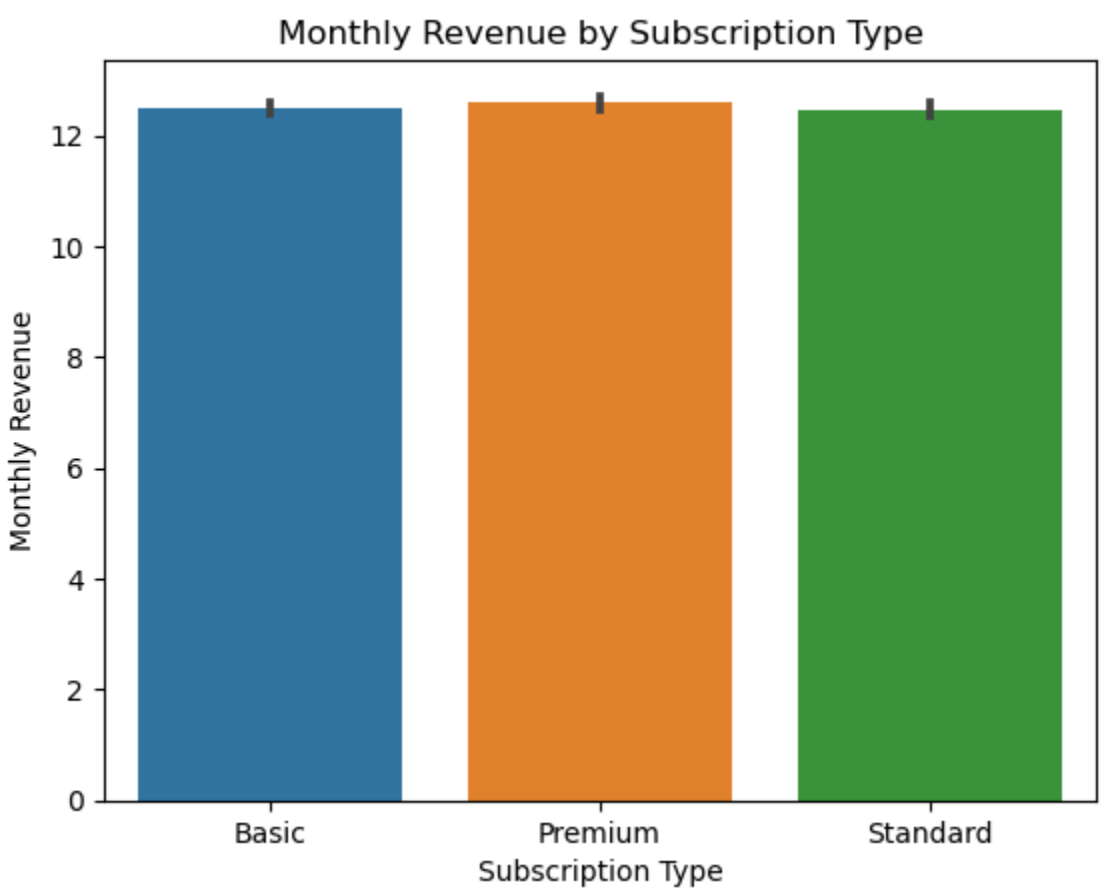
Out[19]:
```
User ID              0
Subscription Type    0
Monthly Revenue      0
Join Date            0
Last Payment Date    0
Country              0
Age                  0
Gender               0
Device               0
Plan Duration        0
dtype: int64
```

In [20]:
```
# Exploratory Analysis
```

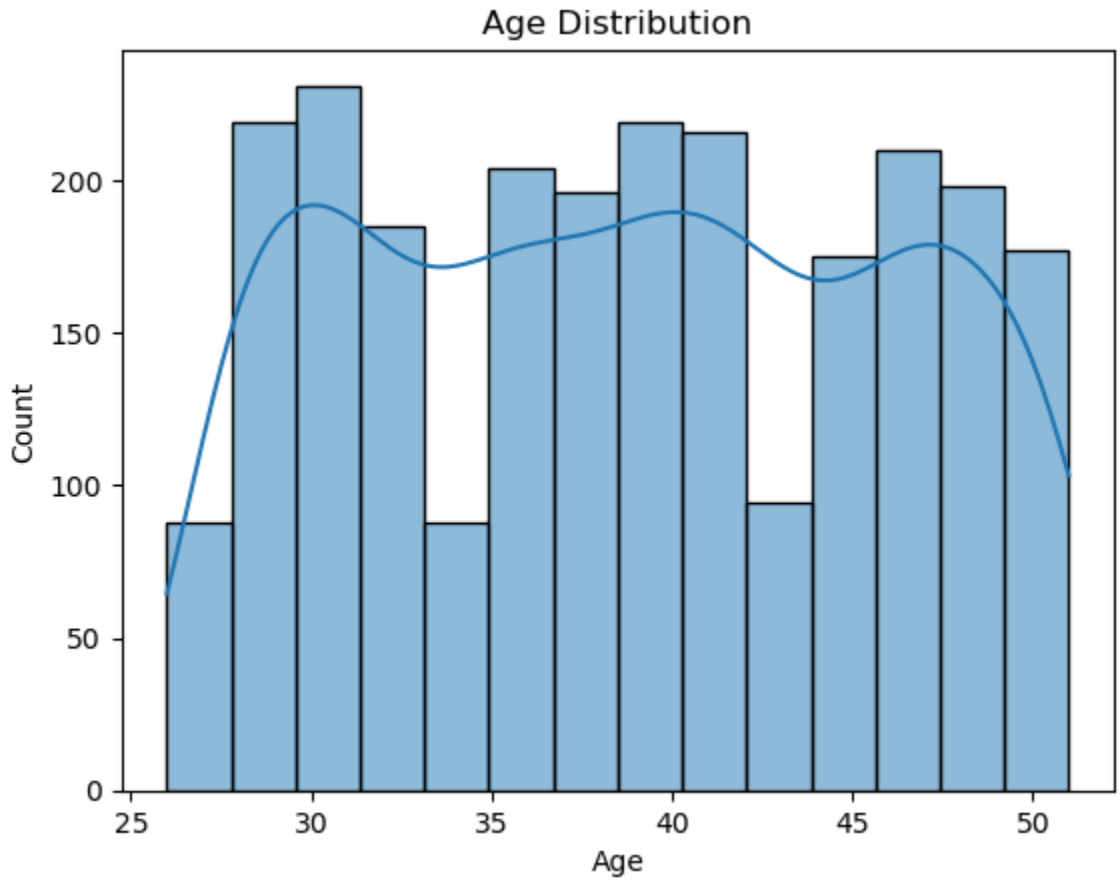
In [21]:
```
# Distribution of subscription types

subscription_counts = df['Subscription Type'].value_counts()
subscription_counts.plot(kind='bar')
plt.title('Distribution of Subscription Types')
plt.xlabel('Subscription Type')
plt.ylabel('Number of Users')
plt.show()
```



In [23]:
```
# Monthly revenue by subscription type

sns.barplot(x='Subscription Type', y='Monthly Revenue', data=df)
plt.title('Monthly Revenue by Subscription Type')
plt.show()
```

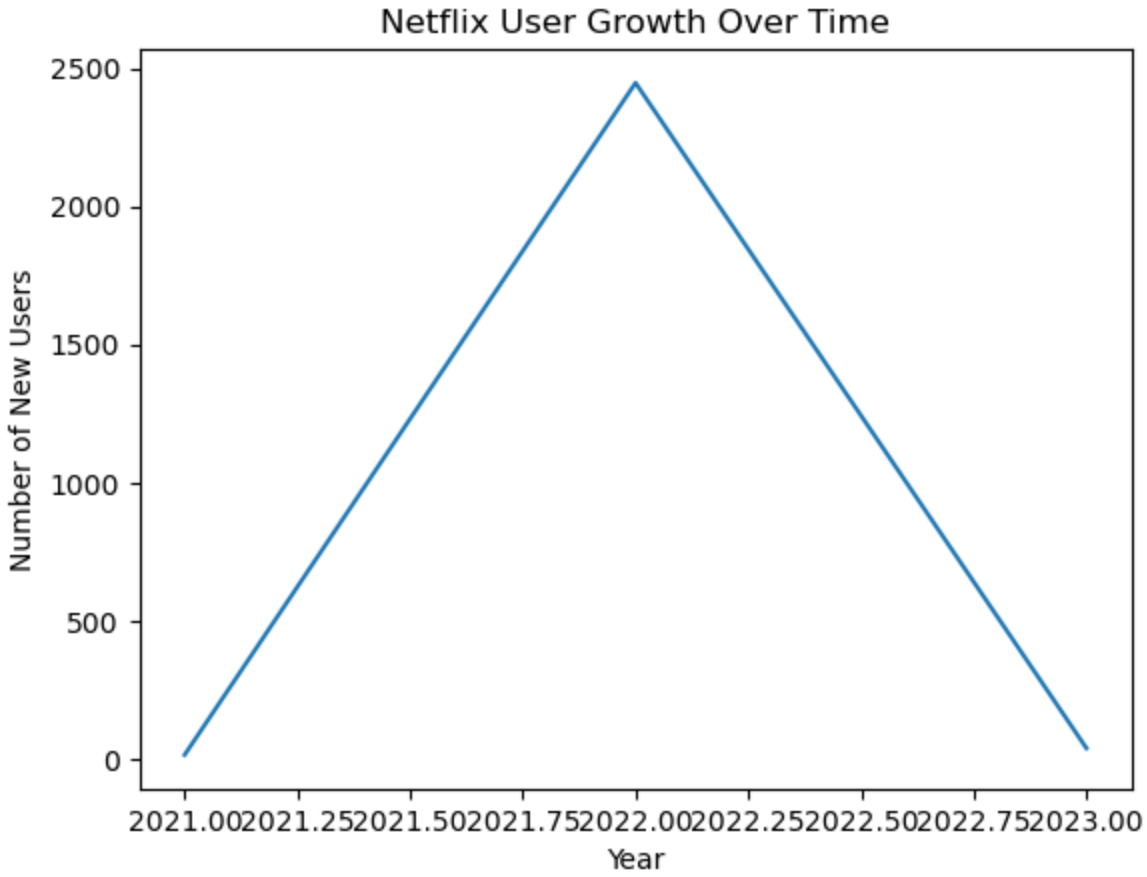


In [24]:
```
# Age distribution

sns.histplot(df['Age'], kde=True)
plt.title('Age Distribution')
plt.show()
```

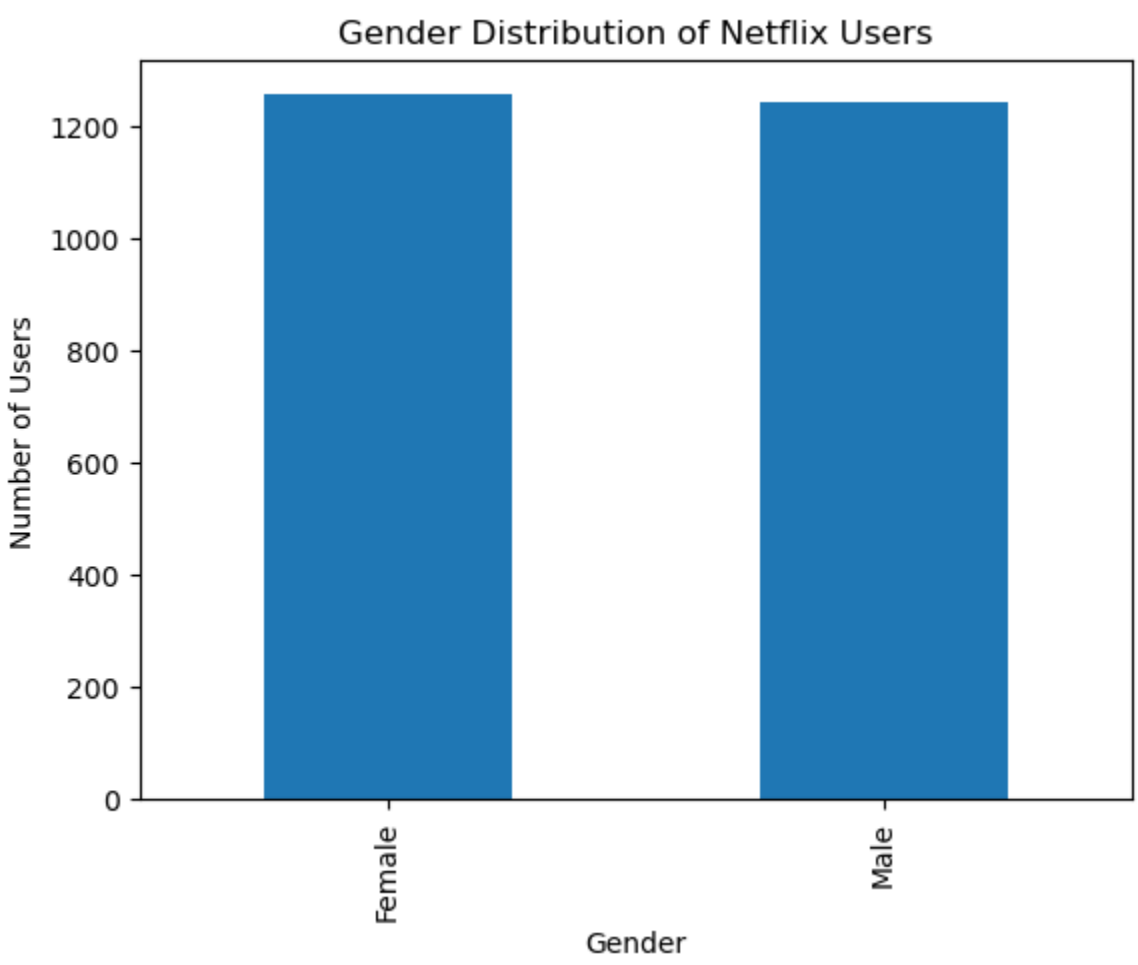


In [25]:
```
# User growth over time
```

```
df['Join Date'] = pd.to_datetime(df['Join Date'])
user_growth = df.groupby(df['Join Date'].dt.year)['User ID'].count()
user_growth.plot(kind='line')
plt.title('Netflix User Growth Over Time')
plt.xlabel('Year')
plt.ylabel('Number of New Users')
plt.show()
```



In [26]:
```
# Gender distribution

gender_counts = df['Gender'].value_counts()
gender_counts.plot(kind='bar')
plt.title('Gender Distribution of Netflix Users')
plt.xlabel('Gender')
plt.ylabel('Number of Users')
plt.show()
```

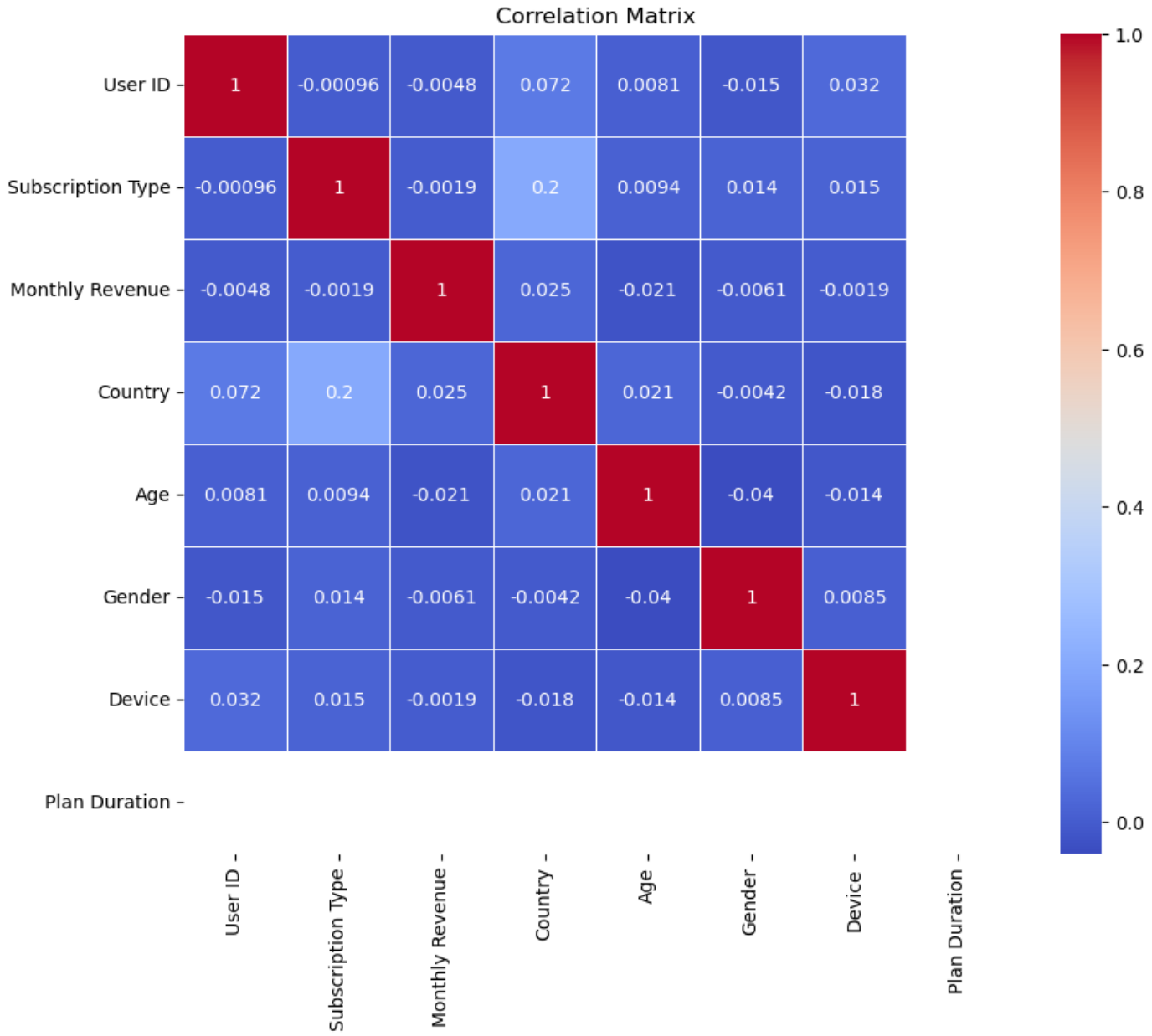


In [27]:
```
# Country with the most users

country_counts = df['Country'].value_counts()
most_popular_country = country_counts.idxmax()
print(f'The country with the most Netflix users is {most_popular_country}.')
```

```
The country with the most Netflix users is United States.
```

In [36]:
```
#Finding Correlation

# Compute the correlation matrix
correlation_matrix = df.corr()

# Create a heatmap to visualize the correlations
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_14504\4113709594.py:4: FutureWarning: The defau
lt value of numeric_only in DataFrame.corr is deprecated. In a future version, it will d
efault to False. Select only valid columns or specify the value of numeric_only to silen
ce this warning.
  correlation_matrix = df.corr()
```



In [37]:
```
 # Encoding Categorical Data using LabelEncoder

categorical_columns = ['Subscription Type', 'Gender', 'Device', 'Country', 'Plan Duratio
encoder = LabelEncoder()

for column in categorical_columns:
    df[column] = encoder.fit_transform(df[column])
```

In [38]:
```
df.head(5)
```

Out[38]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Plan Duration
0	1	0	10	2022-01-15	10-06-23	9	28	1	2	0
1	2	1	15	2021-05-09	22-06-23	2	35	0	3	0
2	3	2	12	2023-02-28	27-06-23	8	42	1	1	0
3	4	2	12	2022-10-07	26-06-23	0	51	0	0	0
4	5	0	10	2023-01-05	28-06-23	4	33	1	2	0

In [41]:
```
#Separating feature and target

X = df.drop(columns=['Monthly Revenue', 'Join Date', 'Last Payment Date'])  # Excluding
y = df['Monthly Revenue']
```

In [42]:
```
X
```

Out[42]:

	User ID	Subscription Type	Country	Age	Gender	Device	Plan Duration
0	1	0	9	28	1	2	0
1	2	1	2	35	0	3	0
2	3	2	8	42	1	1	0
3	4	2	0	51	0	0	0
4	5	0	4	33	1	2	0
...	...	...	...	...	...	...	...
2495	2496	1	7	28	0	1	0
2496	2497	0	7	33	0	1	0
2497	2498	2	9	38	1	0	0
2498	2499	2	2	48	0	3	0
2499	2500	0	9	35	0	1	0

2500 rows × 7 columns

In [44]:
```
y
```

Out[44]:
```
0       10
1       15
2       12
3       12
4       10
        ..
2495    14
2496    15
2497    12
2498    13
2499    15
Name: Monthly Revenue, Length: 2500, dtype: int64
```

In [45]:
```
# Splitting data into training and testing

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

In [46]:
```
# Standardizing data

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [ ]:
```
#Model training and evaluation
```

In [47]:
```
#Linear Regression

lr= LinearRegression()
lr.fit(X_train,y_train)
```

```
print("accuracy on training set:{:.2}".format(lr.score(X_train,y_train)))
print("accuracy on test set:{:.2}".format(lr.score(X_test,y_test)))
```

```
accuracy on training set:0.0023
accuracy on test set:-0.0088
```

In [48]:

```
#Random Forest Regressor

rf = RandomForestRegressor()
rf.fit(X_train,y_train)
print("accuracy on training set:{:.2}".format(rf.score(X_train,y_train)))
print("accuracy on test set:{:.2}".format(rf.score(X_test,y_test)))
```

```
accuracy on training set:0.85
accuracy on test set:-0.064
```

In [53]:

```
#Prediction

input_data = [4, 2, 0, 51, 0, 0, 0]  # Input data as a list
input_data_as_numpy_array = np.array(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)
prediction = rf.predict(input_data_reshaped)
print(prediction)
```

```
[12.56]
```