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

```
df=pd.read_csv('/content/Netflix_Userbase[1].csv')
df
```

	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
[illegible]	[illegible]	[illegible]	[illegible]	09-	[illegible]	United	[illegible]	[illegible]	[illegible]	[illegible]

```
df.dtypes
```

```
User ID               int64
Subscription Type    object
Monthly Revenue       int64
Join Date            object
Last Payment Date    object
Country              object
Age                   int64
Gender               object
Device               object
Plan Duration        object
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   User ID            2500 non-null   int64 
 1   Subscription Type  2500 non-null   object
 2   Monthly Revenue    2500 non-null   int64 
 3   Join Date          2500 non-null   object
 4   Last Payment Date  2500 non-null   object
 5   Country            2500 non-null   object
 6   Age                2500 non-null   int64 
 7   Gender             2500 non-null   object
 8   Device             2500 non-null   object
 9   Plan Duration      2500 non-null   object
dtypes: int64(3), object(7)
memory usage: 195.4+ KB
```

```
df.isna().sum()
```

```
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
```

```
df.shape
df.shape

     (2500, 10)
```

```
df.columns

     Index(['User ID', 'Subscription Type', 'Monthly Revenue', 'Join Date',
            'Last Payment Date', 'Country', 'Age', 'Gender', 'Device',
            'Plan Duration'],
           dtype='object')
```

```
cols=df.columns
cols

     Index(['User ID', 'Subscription Type', 'Monthly Revenue', 'Join Date',
            'Last Payment Date', 'Country', 'Age', 'Gender', 'Device',
            'Plan Duration'],
           dtype='object')
```

```
#Unique items in each column
for cols in df.columns:
  print("Unique elements in", cols, 'is', '\n', df[cols].unique())

      '03-08-22' '20-06-22' '29-03-23' '30-01-22' '18-09-22' '09-11-21'
      '23-12-22' '08-05-23' '26-01-22' '11-10-22' '27-09-22' '14-12-22'
      '28-05-22' '29-12-21' '25-12-22' '18-11-22' '08-11-22' '11-01-23'
      '07-12-22' '13-04-23' '06-04-23' '04-03-22' '28-12-21' '03-10-22'
      '05-01-22' '05-02-22' '19-03-22' '28-02-22' '26-07-22' '05-03-22'
      '07-03-22' '13-01-23' '22-01-23' '17-10-22' '29-03-22' '04-04-22'
      '08-06-22' '24-05-22' '26-01-23' '14-07-22' '01-06-22' '07-05-22'
      '05-04-22' '19-05-22' '30-10-21' '07-08-22' '20-09-22' '18-10-22'
      '03-05-22' '05-11-22' '21-08-22' '21-09-22' '23-11-21' '06-10-22'
      '24-06-22' '18-07-22' '13-09-22' '10-08-22' '24-03-23' '02-05-22'
      '11-05-22' '10-06-22' '31-08-22' '17-08-22' '05-05-22' '27-11-22'
      '17-11-22' '15-12-22' '12-12-22' '23-11-22' '11-03-23' '19-08-22'
      '19-10-22' '21-12-22' '27-10-22' '23-10-22' '19-04-22' '11-03-22'
      '16-04-22' '11-12-22' '12-01-23' '29-04-22' '27-04-22' '03-06-22'
      '23-04-22' '22-04-22' '12-04-22' '23-05-22' '26-09-22' '16-06-22'
      '26-05-22' '12-05-22' '16-07-22' '02-09-22' '08-07-22' '26-06-22'
      '05-08-22' '26-08-22' '09-06-22' '31-10-22' '13-10-22' '04-07-22'
      '31-07-22' '27-07-22' '30-07-22' '06-08-22' '16-09-22' '05-01-23'
      '10-12-22' '22-11-22' '15-11-22' '28-10-22' '08-09-22' '08-10-22'
      '13-11-22' '16-11-22' '26-11-22' '08-01-23' '30-09-22' '04-11-22'
      '30-10-22' '29-10-22' '30-08-22' '17-05-22' '08-05-22' '17-07-22'
      '27-06-22' '31-05-22' '02-06-22' '20-04-22' '15-09-22' '09-08-22'
      '25-06-22' '30-05-22' '21-05-22' '20-05-22' '17-06-22' '13-07-22'
      '25-07-22' '16-10-22' '15-10-22' '04-09-22' '20-08-22' '02-08-22'
      '12-09-22' '12-11-22' '02-11-22' '04-08-22' '01-09-22' '14-10-22'
      '20-11-22' '25-11-22' '04-12-22' '09-12-22' '03-11-22' '11-06-22'
      '06-07-22' '14-06-22' '07-06-22' '27-05-22' '21-04-22' '28-04-22'
      '25-08-22' '22-08-22' '22-06-22' '04-06-22' '22-05-22' '29-05-22'
      '24-07-22' '28-07-22' '11-08-22' '24-08-22' '08-08-22' '16-08-22'
      '12-10-22' '11-11-22' '21-11-22' '08-12-22' '01-11-22' '24-10-22'
      '25-10-22' '22-10-22' '06-09-22' '18-06-22' '12-06-22' '27-08-22'
      '13-08-22' '12-08-22' '11-07-22' '19-07-22' '21-07-22' '03-07-22'
      '23-06-22' '05-06-22' '15-07-22' '23-07-22' '03-09-22' '23-08-22'
      '21-10-22' '04-10-22' '06-11-22' '19-11-22' '10-10-22' '01-07-22'
      '28-06-22' '29-06-22' '18-08-22' '26-10-22' '07-10-22' '17-09-22'
      '09-11-22' '06-06-22' '19-06-22' '02-07-22' '05-09-22' '24-09-22'
      '19-09-22' '28-09-22' '29-09-22' '29-07-22' '05-07-22' '13-06-22'
      '14-08-22' '15-08-22' '23-09-22' '09-10-22' '14-11-22' '25-09-22'
      '07-11-22' '21-06-22' '09-07-22' '02-10-22' '12-07-22' '10-09-22'
      '20-07-22' '11-09-22' '09-09-22' '22-09-22' '07-09-22' '28-08-22']
     Unique elements in Last Payment Date is
      ['10-06-23' '22-06-23' '27-06-23' '26-06-23' '28-06-23' '25-06-23'
      '24-06-23' '23-06-23' '20-06-23' '29-06-23' '30-06-23' '01-07-23'
      '02-07-23' '03-07-23' '04-07-23' '05-07-23' '06-07-23' '07-07-23'
      '08-07-23' '09-07-23' '10-07-23' '11-07-23' '12-07-23' '13-07-23'
      '14-07-23' '15-07-23']
     Unique elements in Country is
      ['United States' 'Canada' 'United Kingdom' 'Australia' 'Germany' 'France'
      'Brazil' 'Mexico' 'Spain' 'Italy']
     Unique elements in Age is
      [28 35 42 51 33 29 46 39 37 44 31 45 48 27 38 36 30 43 32 41 26 34 49 40
      47 50]
     Unique elements in Gender is
      ['Male' 'Female']
     Unique elements in Device is
      ['Smartphone' 'Tablet' 'Smart TV' 'Laptop']
     Unique elements in Plan Duration is
      ['1 Month']
```

```
#Value counts of each columns
for cols in df.columns:
  print("Value count of", cols, 'is', '\n', df[cols].value_counts())
```

```
14-07-23     16
15-07-23      6
20-06-23      1
10-06-23      1
Name: Last Payment Date, dtype: int64
Value count of Country is
 United States     451
Spain             451
Canada            317
United Kingdom    183
Australia         183
Germany           183
France            183
Brazil            183
Mexico            183
Italy             183
Name: Country, dtype: int64
Value count of Age is
 39    116
30    116
28    115
31    115
41    114
47    111
37    107
35    105
29    104
40    103
42    102
48    101
46     99
36     99
49     97
43     94
33     93
51     93
32     92
45     89
38     89
34     88
27     87
44     86
50     84
26      1
Name: Age, dtype: int64
Value count of Gender is
 Female    1257
Male      1243
Name: Gender, dtype: int64
Value count of Device is
 Laptop        636
Tablet        633
Smartphone    621
Smart TV      610
Name: Device, dtype: int64
Value count of Plan Duration is
 1 Month    2500
Name: Plan Duration, dtype: int64
```

```
#Converting last payment date to datetime data type
df['Last Payment Date'] = pd.to_datetime(df['Last Payment Date'])
df['Join Date'] = pd.to_datetime(df['Join Date'])
```

```
#No need of column user id for visualization, so dropped it
df.drop(['User ID'], axis=1,inplace=True)
df
```

	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Plan Duration
0	Basic	10	2022-01-15	2023-10-06	United States	28	Male	Smartphone	1 Month
2	Standard	12	2023-02-28	2023-06-27	United Kingdom	42	Male	Smart TV	1 Month

VISUALIZATION'S

```
#Correlation of the dataset
corr=df.corr()
corr
```

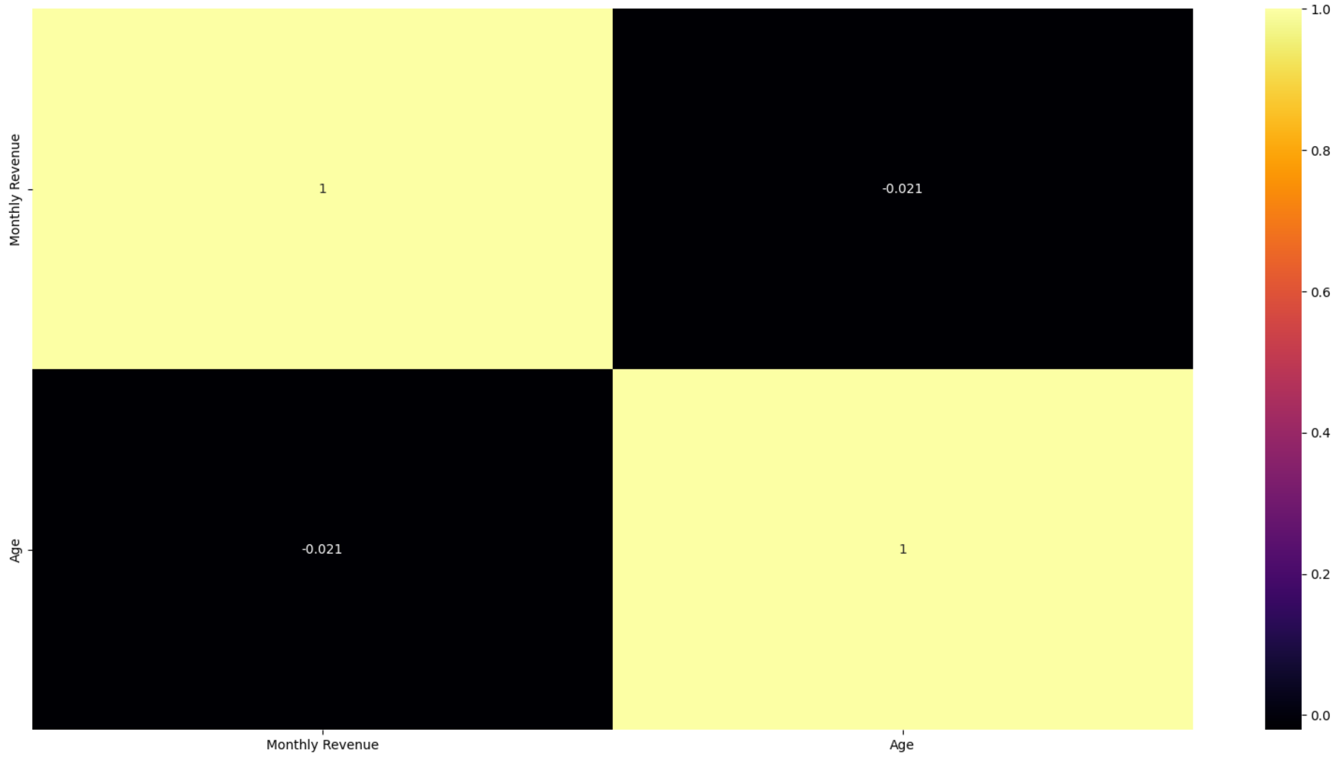
```
<ipython-input-432-78a93a680fcb>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future
  corr=df.corr()
```

	Monthly Revenue	Age
Monthly Revenue	1.000000	-0.021143
Age	-0.021143	1.000000

2499	Basic	15	2022-08-13	2023-12-07	United States	35	Female	Smart TV	1 Month

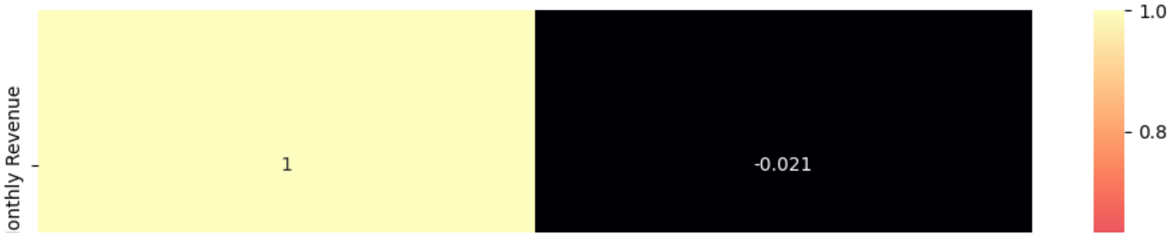
```
#Heat map of the correlation
plt.figure(figsize=(20,10))
sns.heatmap(corr, annot=True, cmap='inferno')
```

```
<Axes: >
```



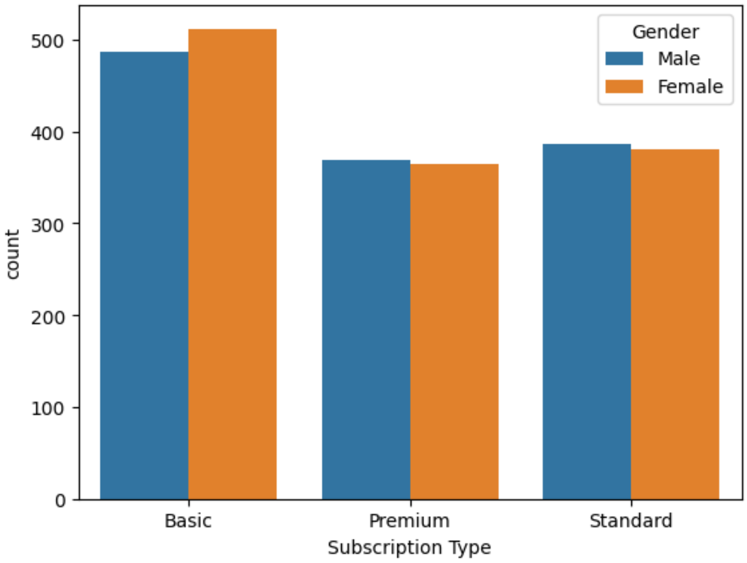
```
corr1=df[['Monthly Revenue','Age']].corr()
plt.figure(figsize=(12,6))
sns.heatmap(corr1, annot=True, cmap='magma')
```

```
<Axes: >
```



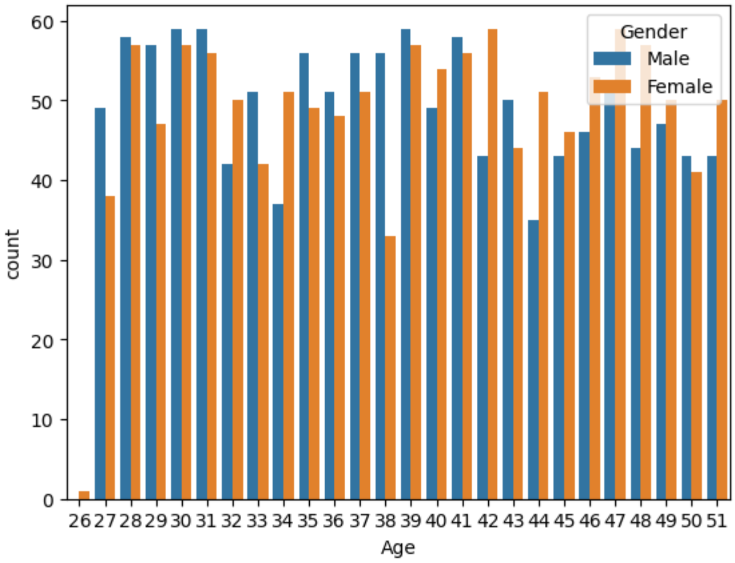
```
#subscription Type based on gender
sns.countplot(x='Subscription Type',hue='Gender',data=df)
```

```
<Axes: xlabel='Subscription Type', ylabel='count'>
```

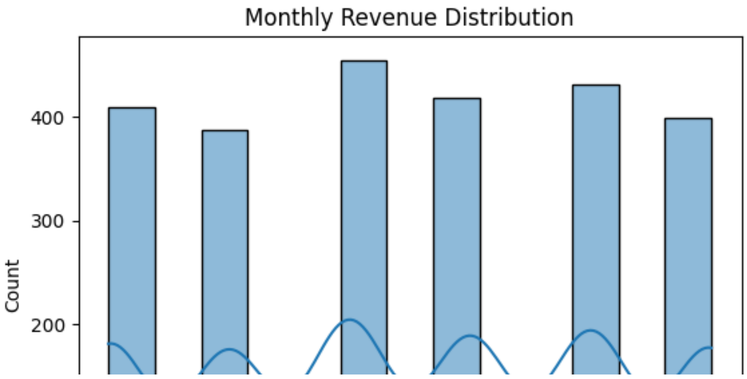


```
#Age based on gender
sns.countplot(x='Age',hue='Gender',data=df)
```

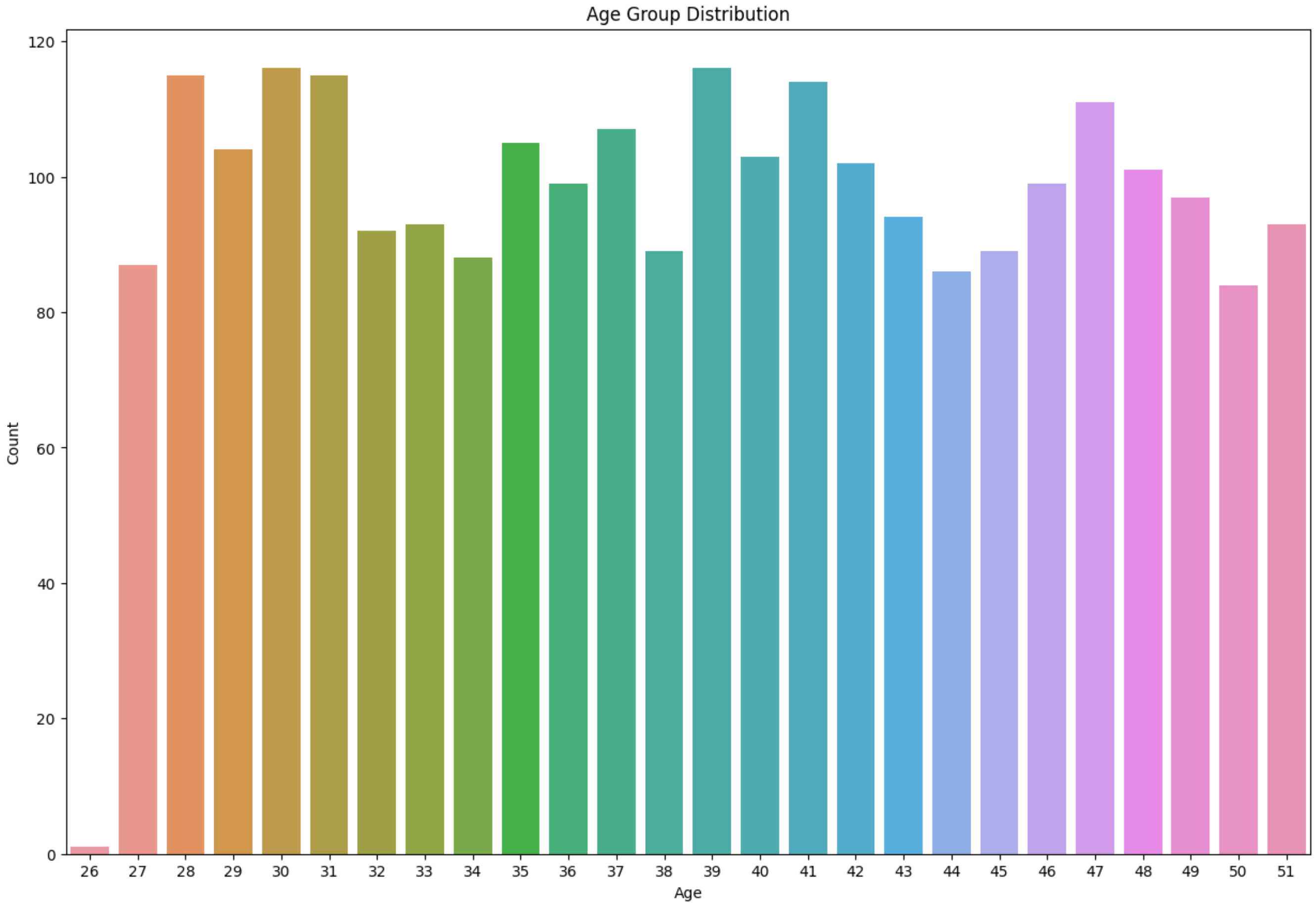
```
<Axes: xlabel='Age', ylabel='count'>
```



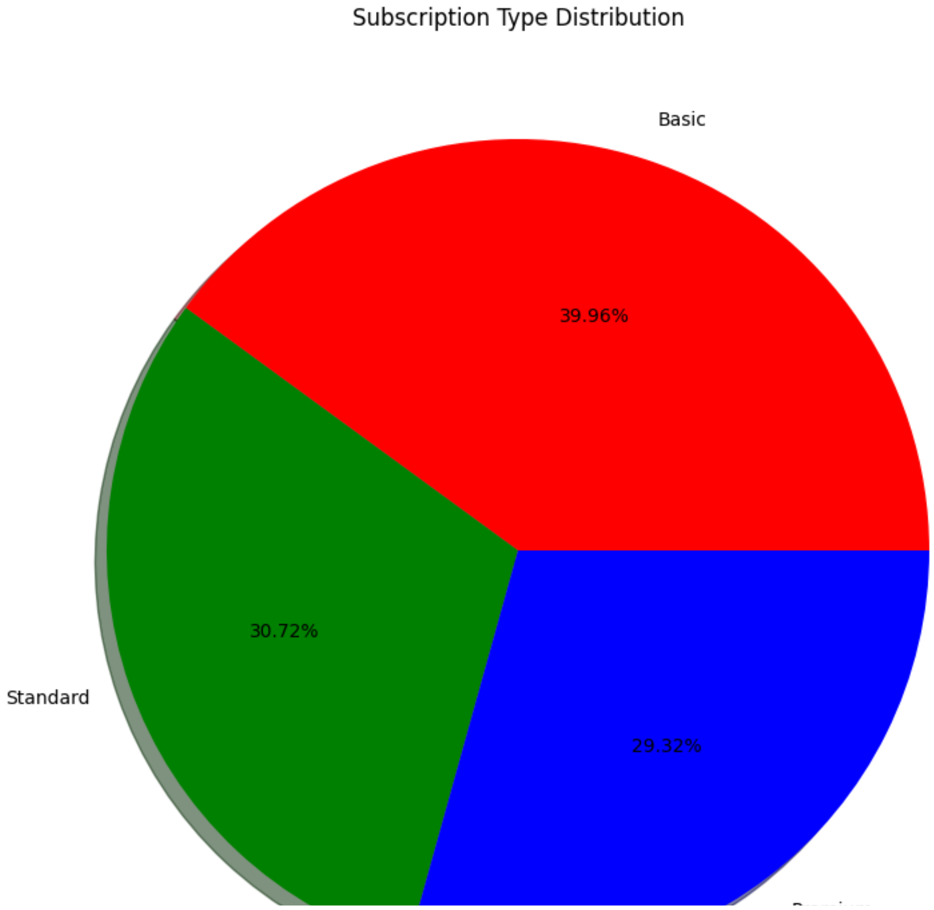
```
#Histogram plot for Monthly revenue
sns.histplot(df["Monthly Revenue"], kde=True)
plt.title("Monthly Revenue Distribution")
plt.show()
```



```
#Age group distribution
plt.figure(figsize=(15,10))
sns.countplot(data=df, x='Age')
plt.title('Age Group Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



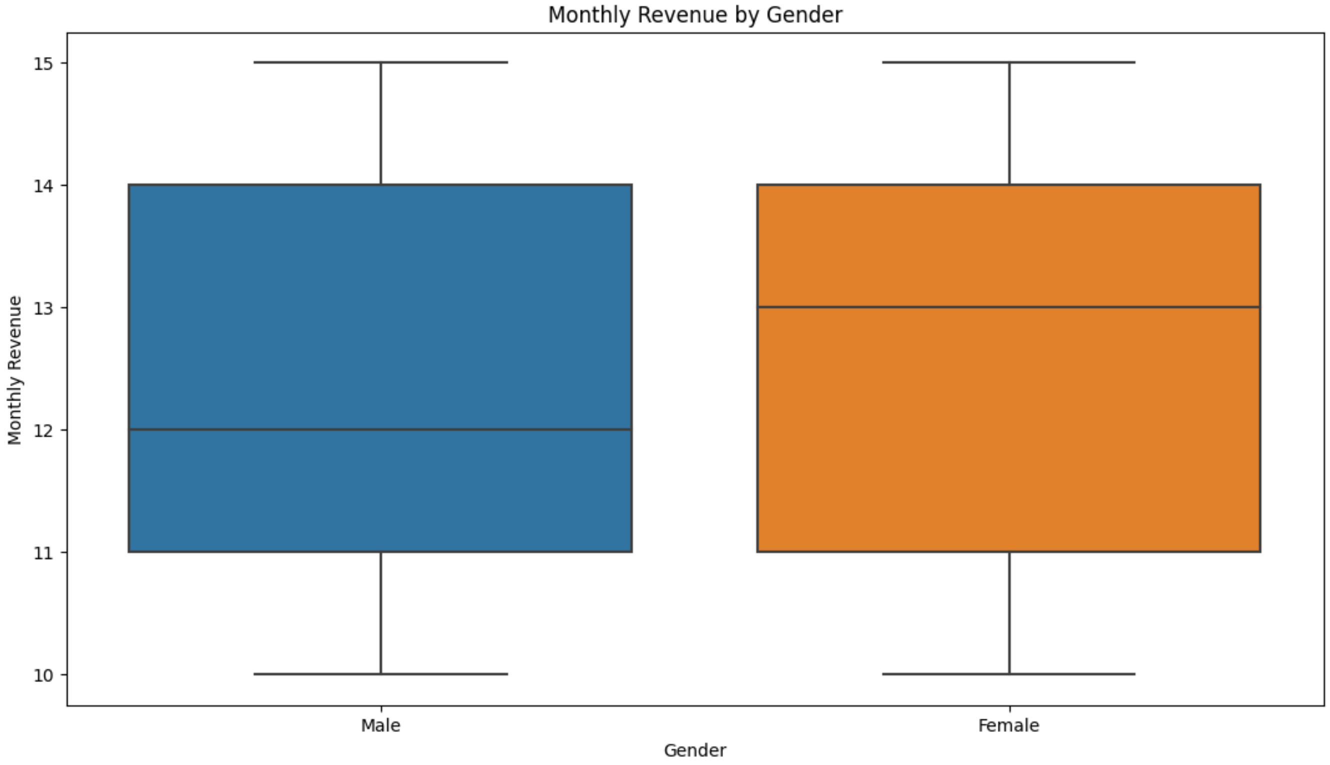
```
#Different susciption type's
plt.figure(figsize=(20,10))
sub_types=df["Subscription Type"].value_counts()
plt.pie(sub_types, labels= sub_types.index, autopct= '%1.2f%%', explode=[0,0,0], shadow=True, colors=['red','green','blue'])
plt.title("Subscription Type Distribution")
plt.show()
```



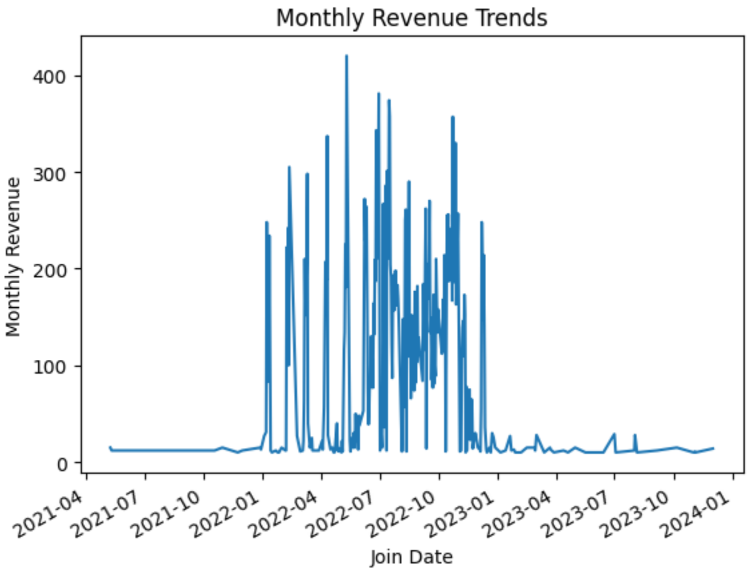
```
#Types of device
plt.figure(figsize=(20,10))
device_types=df["Device"].value_counts()
plt.pie(device_types, labels= device_types.index, autopct= '%1.2f%%', explode=[0,0,0,0], shadow=True, colors=['blue','red','orange','yel
plt.title("Device Types")
plt.show()
```

Device Types

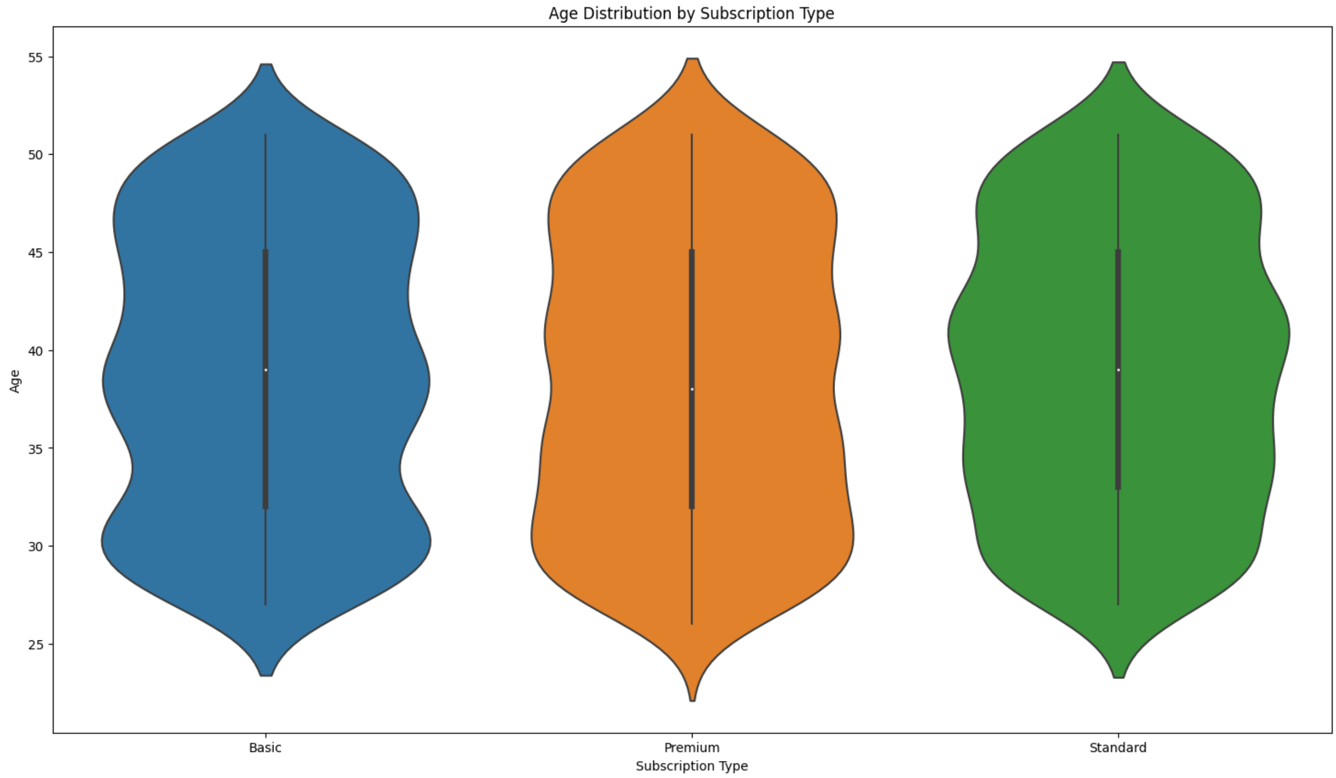
```
#Monthly revenue vs gender
plt.figure(figsize=(13,7))
sns.boxplot(x='Gender', y='Monthly Revenue', data=df)
plt.title('Monthly Revenue by Gender')
plt.xlabel('Gender')
plt.ylabel('Monthly Revenue')
plt.show()
```



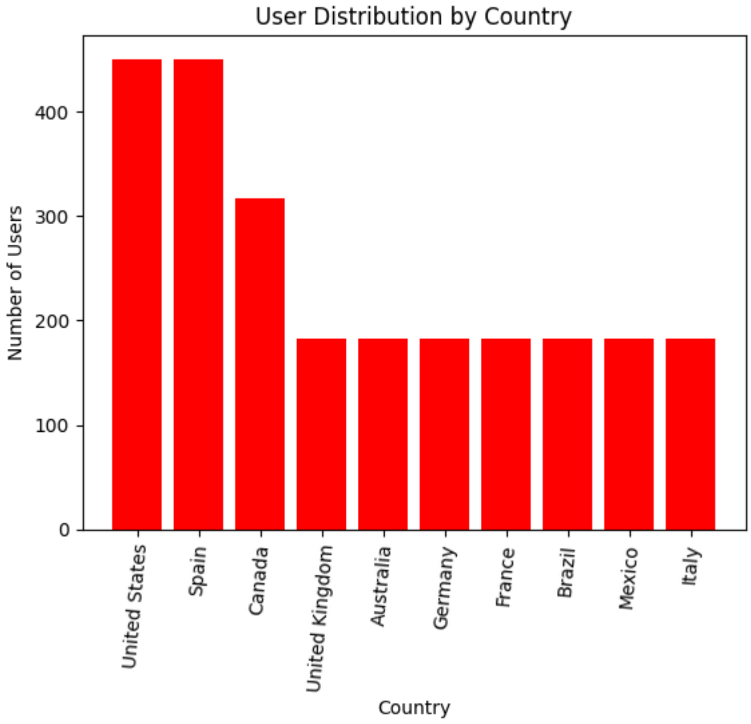
```
#Revenue Analysis
revenue_trends= df.groupby("Join Date")["Monthly Revenue"].sum()
revenue_trends.plot(kind="line")
plt.title("Monthly Revenue Trends")
plt.xlabel("Join Date")
plt.ylabel("Monthly Revenue")
plt.show()
```



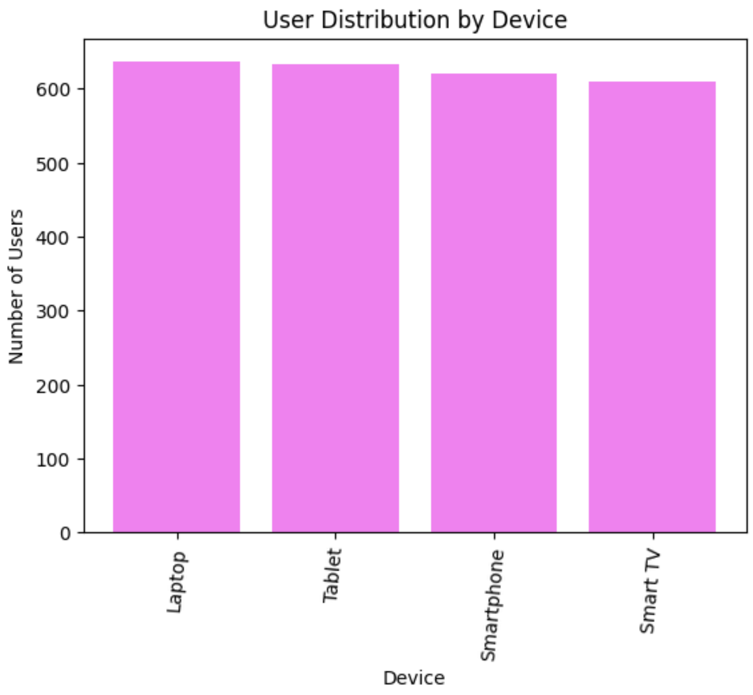
```
#Age distribution by subscription type
plt.figure(figsize=(18,10))
sns.violinplot(x='Subscription Type', y='Age', data=df)
plt.title('Age Distribution by Subscription Type')
plt.xlabel('Subscription Type')
plt.ylabel('Age')
plt.show()
```



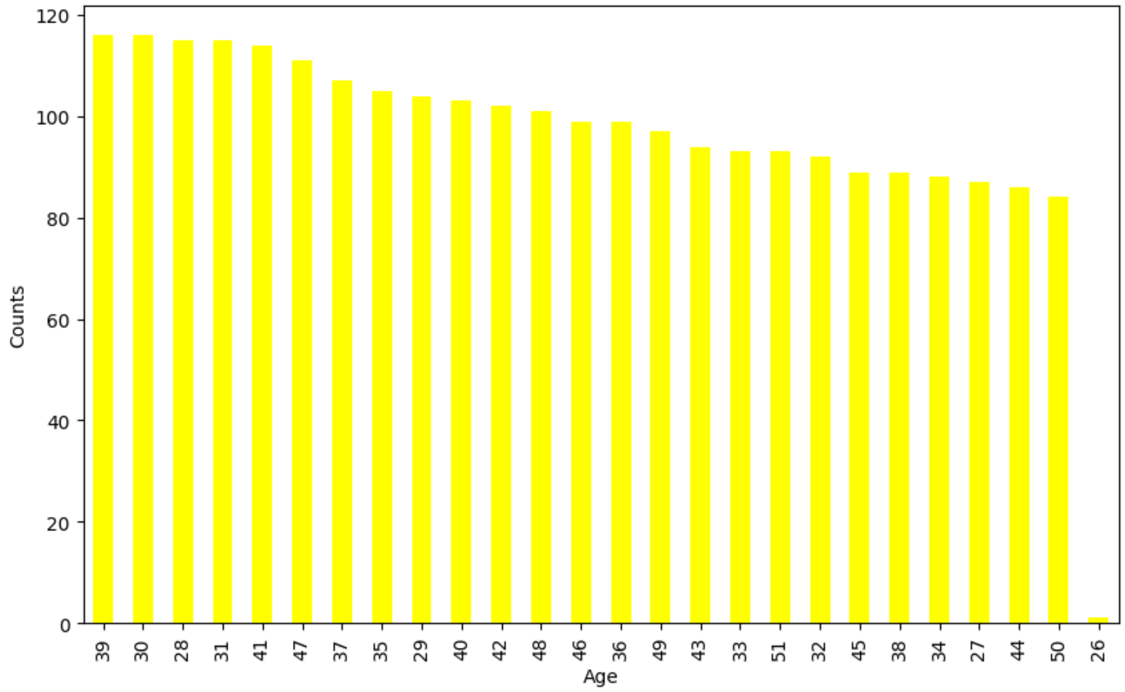
```
#User distribution by countries
country_counts=df["Country"].value_counts()
plt.bar(country_counts.index, country_counts.values, color='red')
plt.title("User Distribution by Country")
plt.xlabel("Country")
plt.ylabel("Number of Users")
plt.xticks(rotation=85)
plt.show()
```



```
#User distribution by device's
device_counts=df["Device"].value_counts()
plt.bar(device_counts.index, device_counts.values, color='violet')
plt.title("User Distribution by Device")
plt.xlabel("Device")
plt.ylabel("Number of Users")
plt.xticks(rotation=85)
plt.show()
```

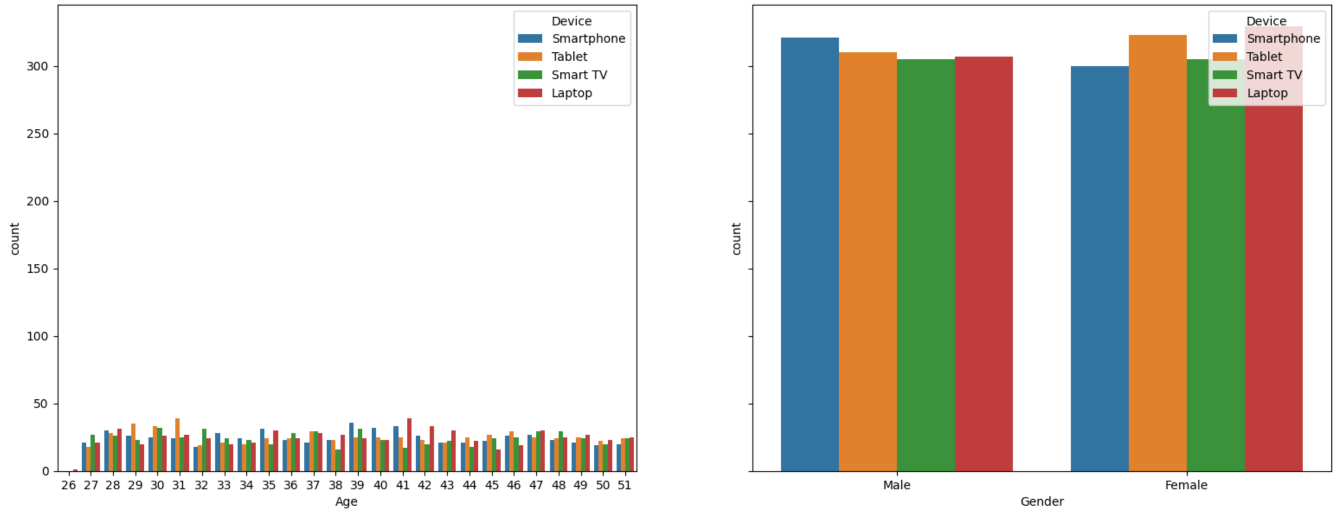


```
#Age counts
df.Age.value_counts().plot(kind= 'bar', figsize=(10,6), color= 'yellow')
plt.ylabel("Counts")
plt.xlabel("Age")
plt.show()
```



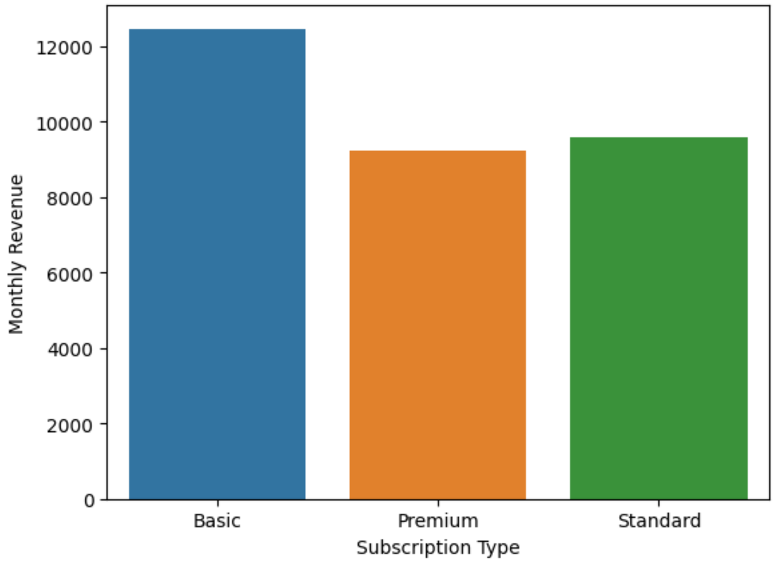
```
#Type of device used by different age category and device used by each gender
fig,ax=plt.subplots(1,2,sharey=True,figsize=(19,7))
sns.countplot(x='Age',hue='Device',data=df, ax=ax[0])
sns.countplot(x='Gender',hue='Device',data=df, ax=ax[1])
```

```
<Axes: xlabel='Gender', ylabel='count'>
```



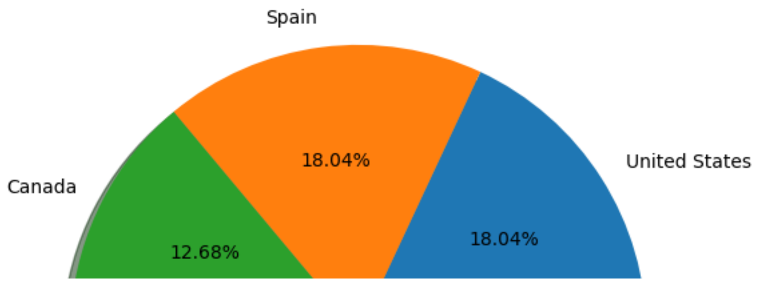
```
#Which suscription type generates more revenue?
st=df.groupby('Subscription Type').sum().reset_index()
sns.barplot(data=st ,x='Subscription Type', y='Monthly Revenue')
plt.show()
```

```
<ipython-input-448-f87c401962db>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a fu
  st=df.groupby('Subscription Type').sum().reset_index()
```



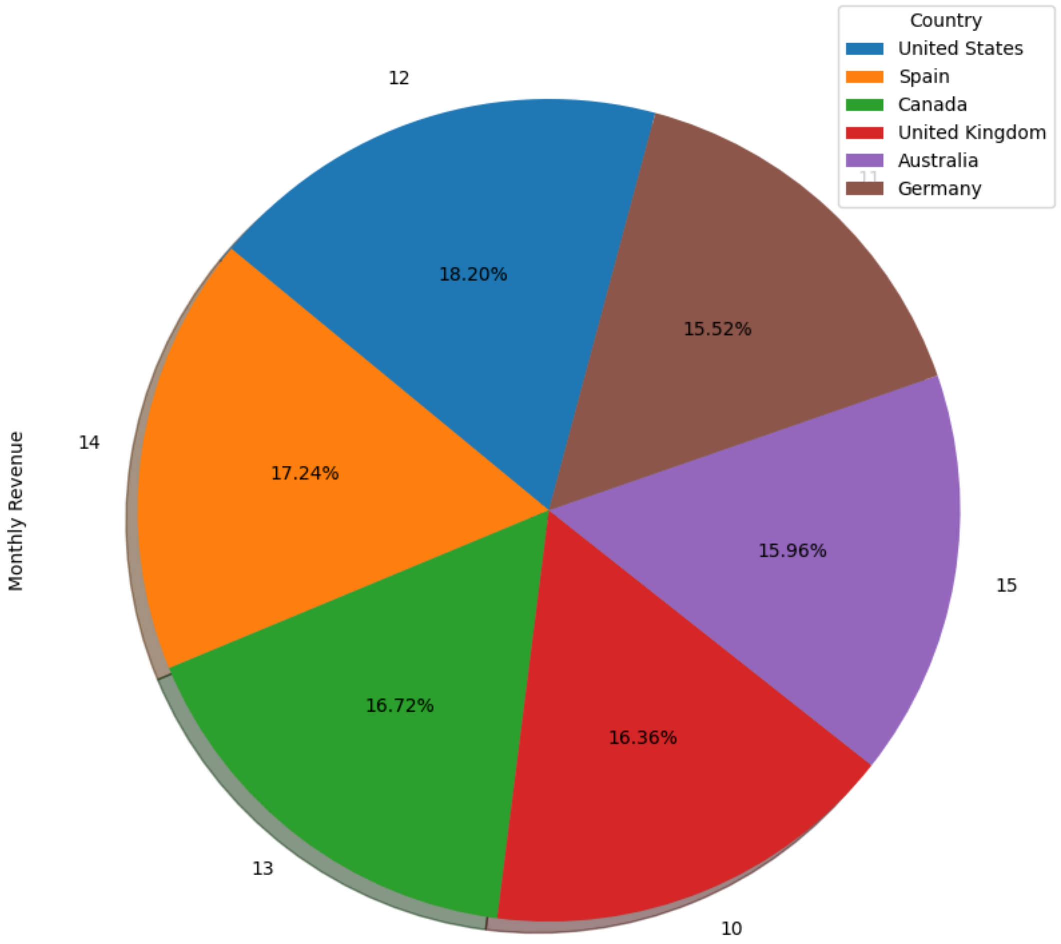
Basic type subscription has maximum purchasing

```
#From which countries more customers are there?
plt.figure(figsize= (17,7))
countries=df.Country.value_counts()
plt.pie(countries,labels= countries.index, shadow='True', autopct='%1.2f%%')
plt.show()
```



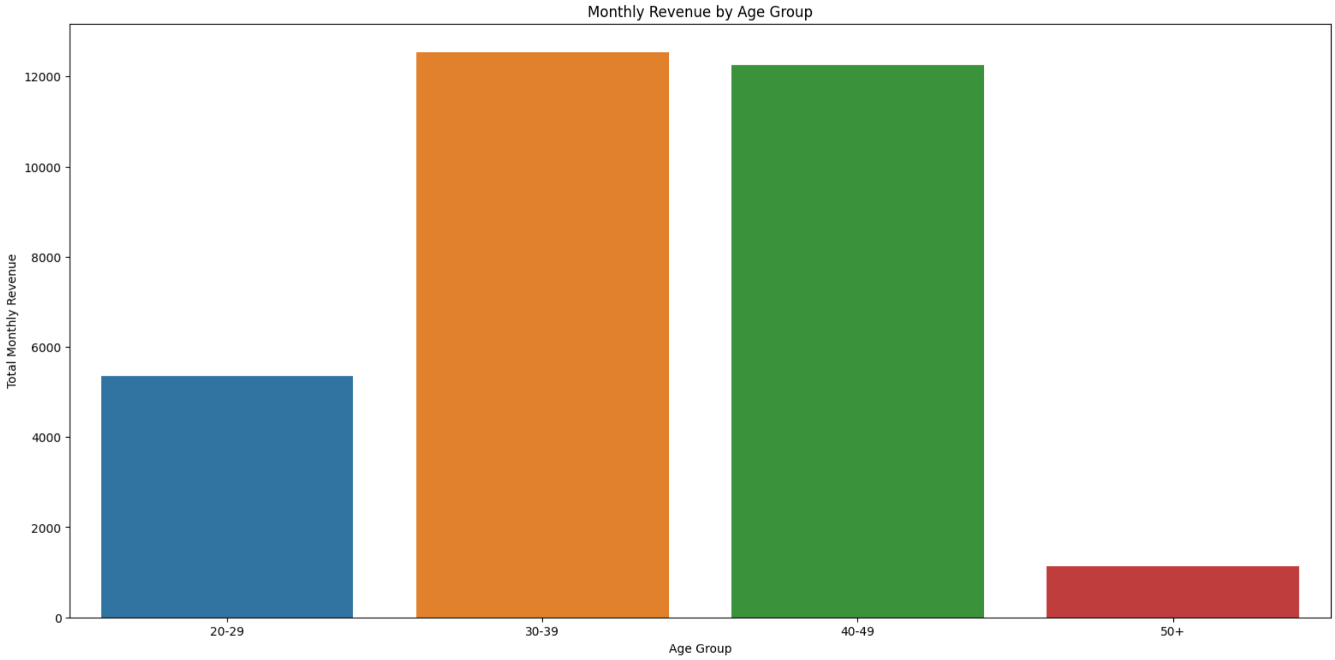
More users are from Spain and US

```
#Countries generating high monthly revenue
country_counts=df['Country'].value_counts()
df['Monthly Revenue'].value_counts().plot(kind='pie',shadow='True' ,autopct='%1.2f%%', startangle = 75,figsize=(18,10))
plt.legend(title= 'Country', labels= country_counts.index, loc='upper right' )
plt.show()
```



Highest revenue is generated by US

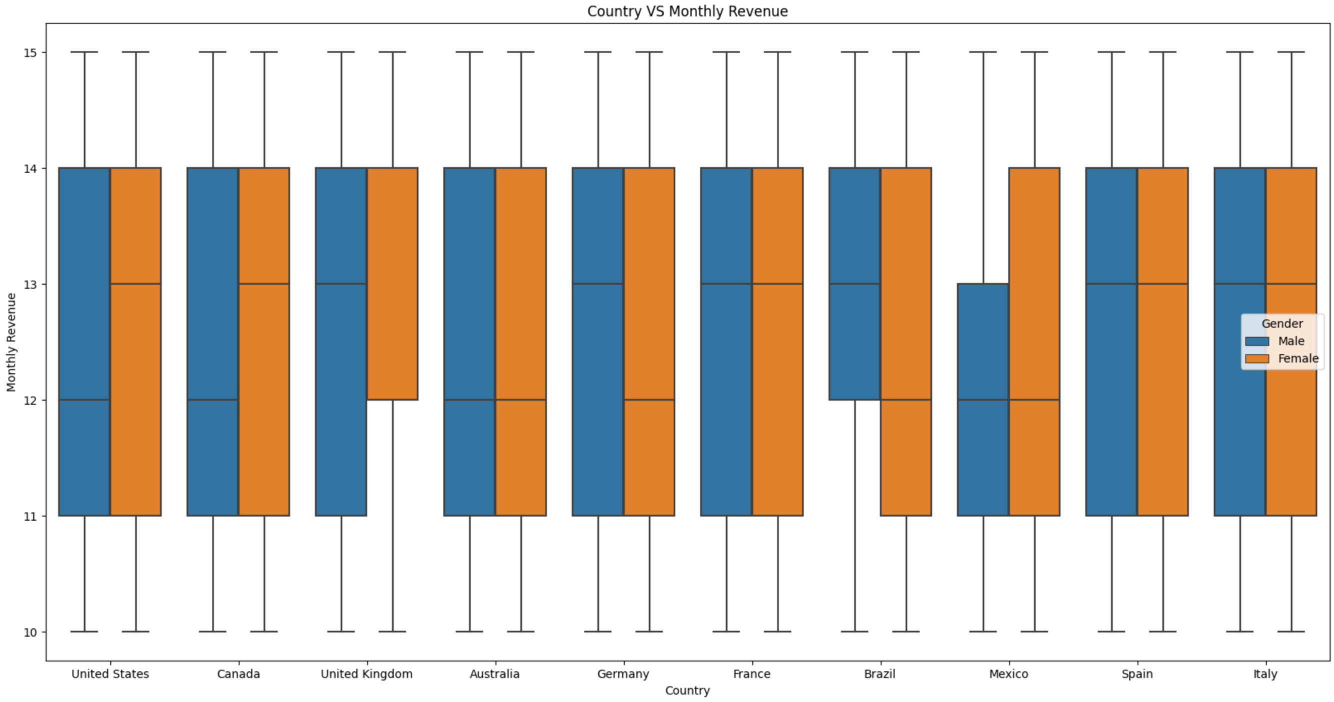
```
#Monthly revenue by age groups
plt.figure(figsize=(19,9))
age_bins= [20, 30, 40, 50, 60]
age_labels= ['20-29', '30-39', '40-49', '50+']

df['age group']= pd.cut(df['Age'], bins=age_bins, labels=age_labels)
revenue_by_age= df.groupby('age group')['Monthly Revenue'].sum().reset_index()
sns.barplot(data=revenue_by_age, x='age group', y='Monthly Revenue')
plt.xlabel('Age Group')
plt.ylabel('Total Monthly Revenue')
plt.title('Monthly Revenue by Age Group')
plt.show()
```



Double-click (or enter) to edit

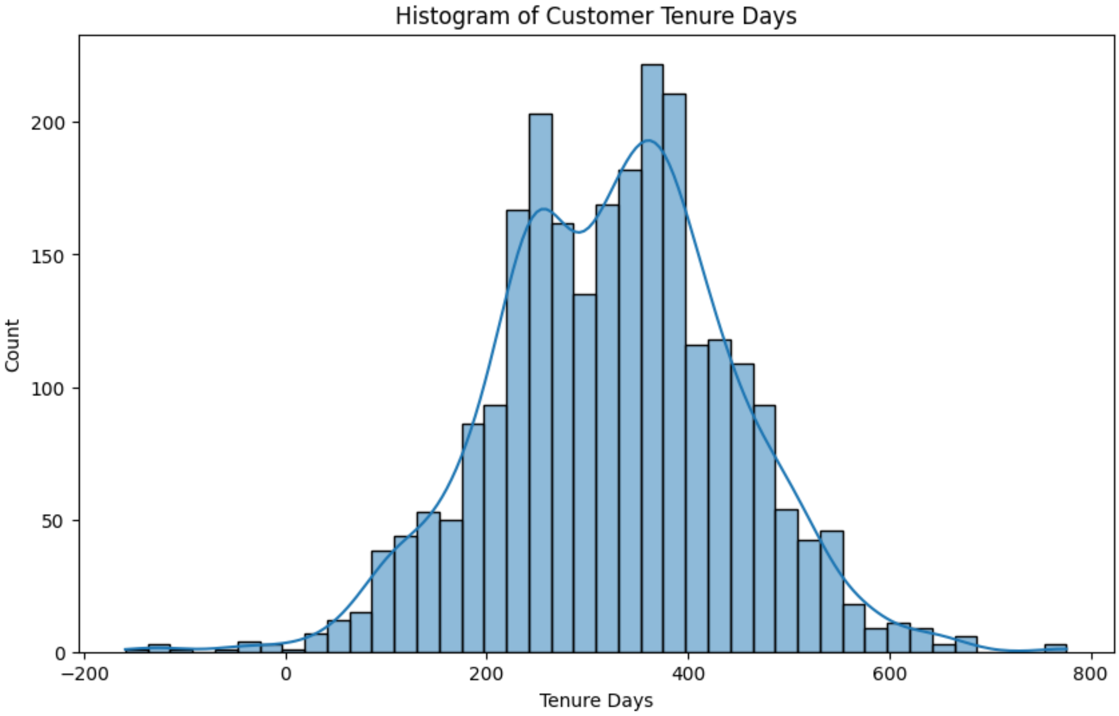
Age group 30-49 doing the most subscription and giving more revenue

```
#Country vs monthly revenue
plt.figure(figsize=(20,10))
sns.boxplot(data=df, x='Country', y='Monthly Revenue', hue= 'Gender')
plt.title("Country VS Monthly Revenue")
plt.xlabel("Country")
plt.ylabel("Monthly Revenue")
plt.show()
```

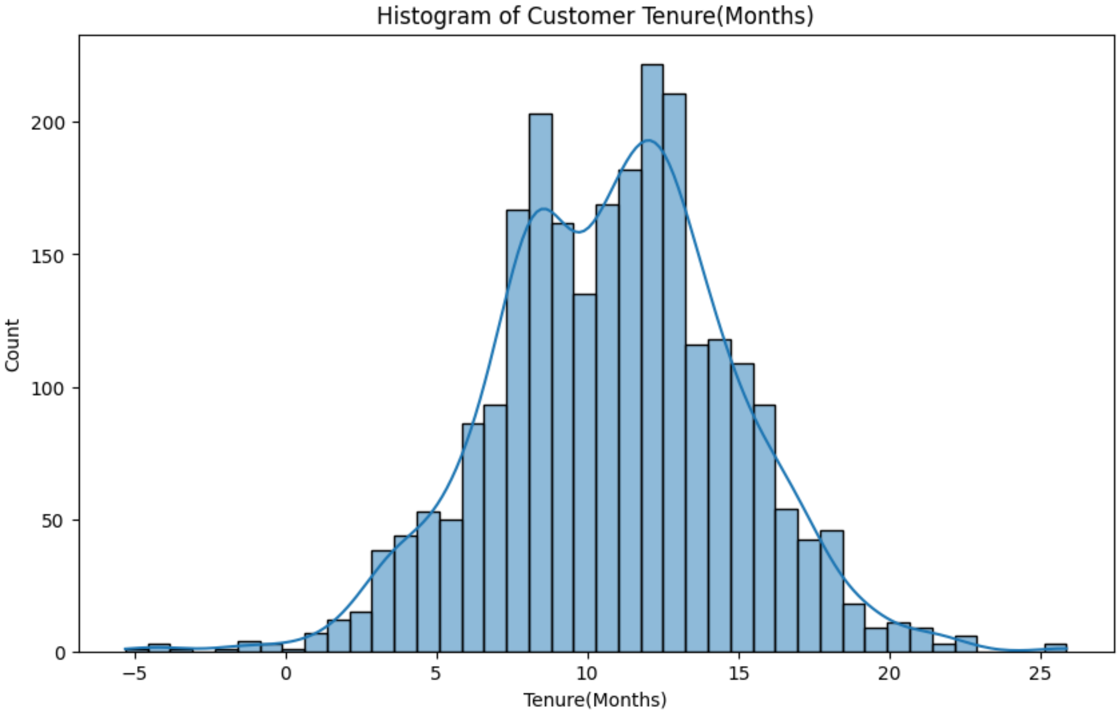


```
#The 'Customer Tenure' column will contain the number of days each user has been associated with the service from their 'Join Date' to t
df['Tenure Days']= (df['Last Payment Date']-df['Join Date']).dt.days #dt.days-is used to extract the number of days from the resulting t
df['Tenure(Months)']=df['Tenure Days']/30
```

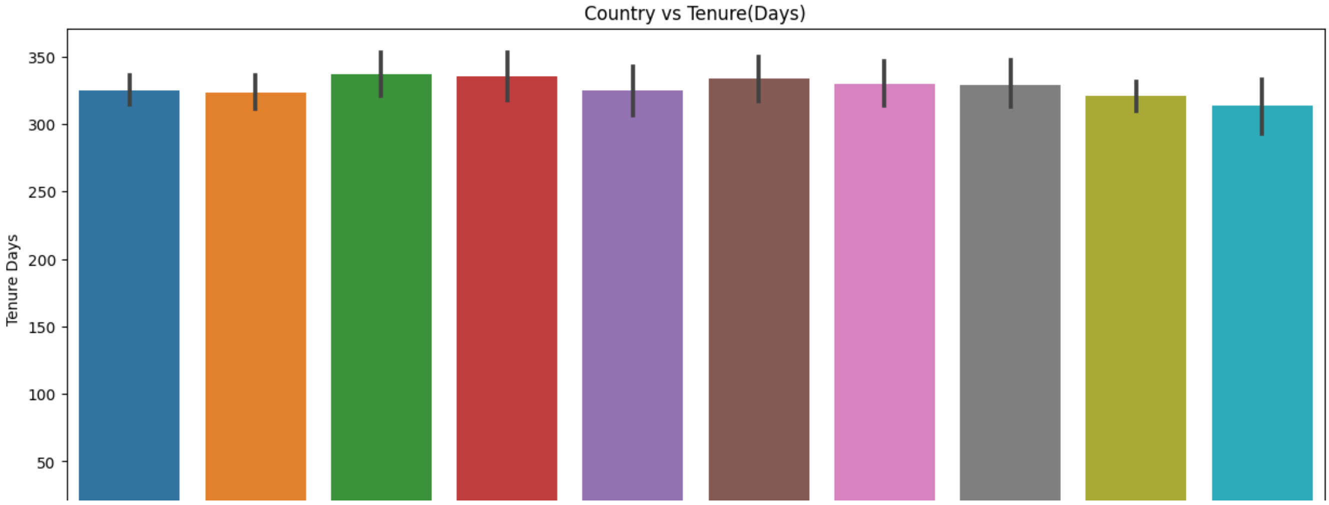
```
plt.figure(figsize=(10,6))
sns.histplot(data=df,x='Tenure Days', kde=True)
plt.title('Histogram of Customer Tenure Days')
plt.show()
```



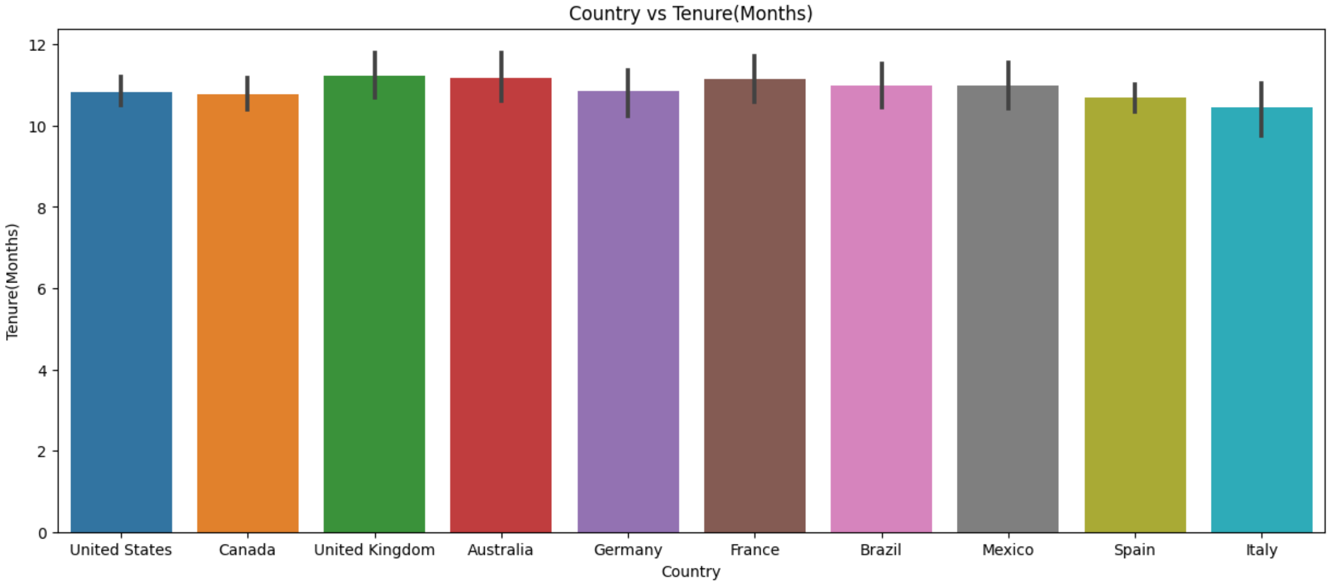
```
plt.figure(figsize=(10,6))
sns.histplot(data=df,x='Tenure(Months)', kde=True)
plt.title('Histogram of Customer Tenure(Months)')
plt.show()
```



```
plt.figure(figsize=(15,6))
sns.barplot(data=df,x='Country',y='Tenure Days')
plt.title('Country vs Tenure(Days)')
plt.show()
```

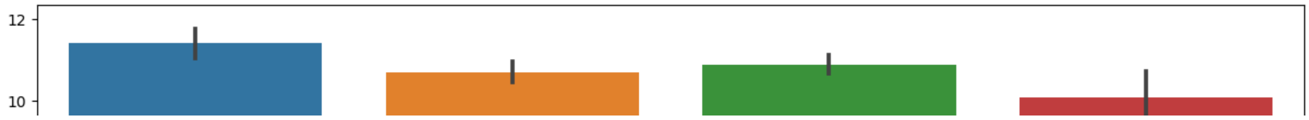


```
plt.figure(figsize=(15,6))
sns.barplot(data=df,x='Country',y='Tenure(Months)')
plt.title('Country vs Tenure(Months)')
plt.show()
```



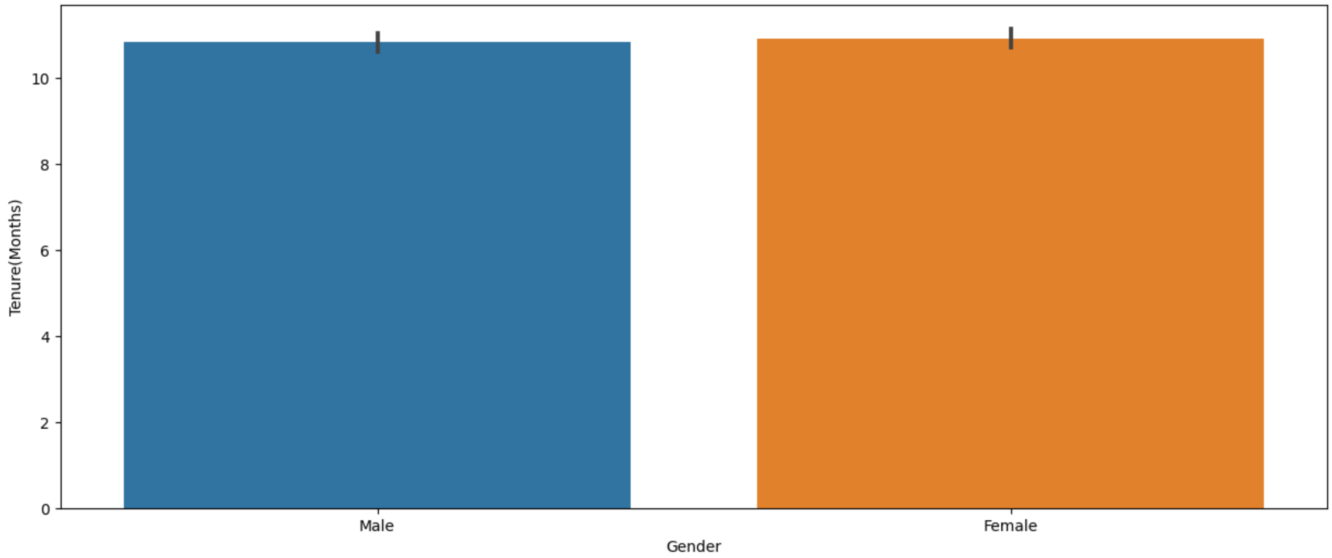
Users in US is generating high Tenure months

```
#Tenure months vs Age group
plt.figure(figsize=(15,6))
sns.barplot(data=df,x='age group',y='Tenure(Months)')
plt.show()
```



Age group 20-29 spend more time on Netflix

```
#Tenure months vs Gender
plt.figure(figsize=(15,6))
sns.barplot(data=df,x='Gender',y='Tenure(Months)')
plt.show()
```



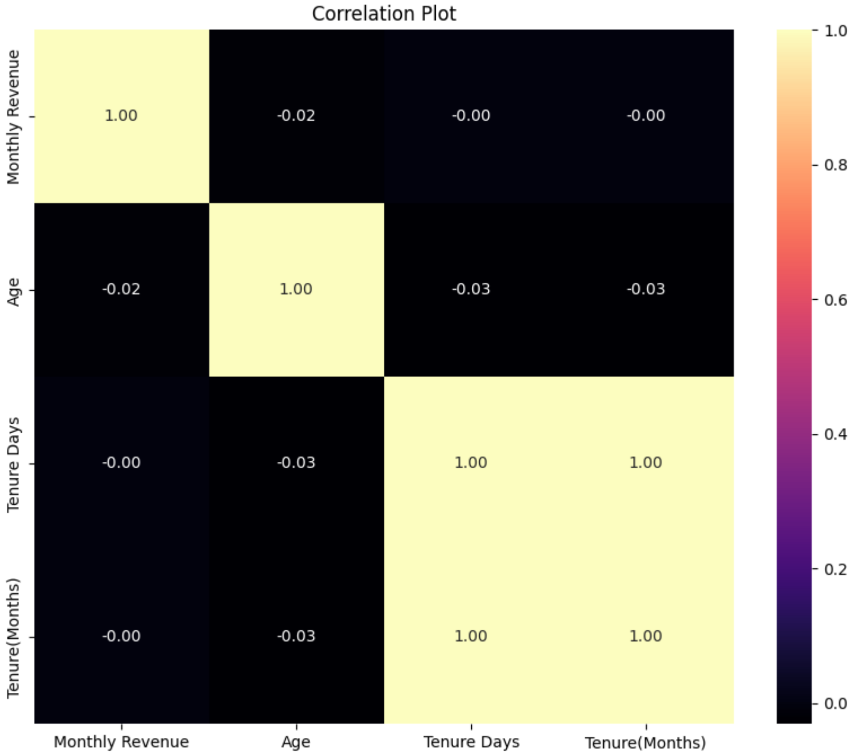
```
new_corr= df.corr()
new_corr
```

```
<ipython-input-460-381bdbc5009b>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future
  new_corr= df.corr()
```

	Monthly Revenue	Age	Tenure Days	Tenure(Months)
Monthly Revenue	1.000000	-0.021143	-0.004620	-0.004620
Age	-0.021143	1.000000	-0.031335	-0.031335
Tenure Days	-0.004620	-0.031335	1.000000	1.000000
Tenure(Months)	-0.004620	-0.031335	1.000000	1.000000

```
corr_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(new_corr, annot=True, cmap="magma", fmt=".2f")
plt.title("Correlation Plot")
plt.show()
```

```
ipython-input-461-11b0b21d05aa>:1: FutureWarning: The default value of numeric_only i
 corr_matrix = df.corr()
```



✓ 0s completed at 8:17 PM