N VAMSI

AP21110010844

BATCH 99

Netflix Userbase Data Analysis

Abstract

- This Project focuses on analyzing Netflix user data and subscription patterns to gain insights into customer behavior
- The dataset includes information on user ID, subscription type, monthly revenue, join date, last payment date, country, age, gender, device, and plan duration.
- The main objectives of this analysis are to understand user preferences, identify trends in subscription adoption, and visualize key metrics for better decision-making.

Introduction

- Netflix, a leading online streaming platform, has revolutionized the way people consume entertainment content. With millions of subscribers worldwide, it has become essential to understand user behavior and subscription patterns to optimize service offerings and enhance customer satisfaction.
- This Project aims to analyze a comprehensive dataset containing key attributes such as user ID, subscription type, monthly revenue, join date, last payment date, country, age, gender, device, and plan duration.
- By conducting in-depth analysis and visualizations, we seek to uncover valuable insights that can assist Netflix in making informed decisions for its business growth.

System Requirements

Operating System:

- Windows 10, macOS, or Linux (Ubuntu, CentOS, etc.)

Software:

- Python (3.7 or higher) with necessary libraries: Pandas, NumPy, Matplotlib, Seaborn, Plotly, etc.
- Jupyter Notebook or any preferred integrated development environment (IDE) for Python.
- SQL database management system (DBMS) for data storage and retrieval.

Hardware:

- Processor: Intel Core i5 or equivalent AMD processor (or higher).
- RAM: 8GB (or higher) for smooth data handling and analysis.
- Storage: At least 50GB of free space for storing the dataset and analysis results.
- Graphics Card: A dedicated graphics card is not strictly required for this analysis, but it can enhance the performance of data visualization, especially for large datasets.

Uses of Data Analysis Library

- The combination of NumPy, pandas, scikit-learn, Matplotlib, and Seaborn can provide a comprehensive toolkit for data analysis project. Here are some specific uses of each library within a project:

1.NumPy:

- NumPy provides powerful numerical operations and arrays, making it useful for mathematical computations and manipulation of large datasets.
- It offers essential functions for array manipulation, such as reshaping, slicing, indexing, and aggregations.

2.Pandas:

- It enables efficient handling and preprocessing of structured data, including CSV, Excel, SQL databases, and other formats.
- pandas allows data cleaning, filtering, sorting, merging, and aggregating, making it suitable for data wrangling tasks.

3.scikit-learn:

- scikit-learn offers various preprocessing methods, feature selection techniques, Scaling techniques.

- scikit-learn integrates seamlessly with NumPy and pandas, allowing easy data preparation and model training.

4. Matplotlib:

- It provides a wide range of plot types, including line plots, scatter plots, bar plots, histograms, heatmaps, and more.
- Matplotlib is highly compatible with NumPy and pandas, enabling visualization of data from these libraries.

5.Seaborn:

- Seaborn is a statistical data visualization library that works closely with Matplotlib and enhances its capabilities.
- Seaborn includes functions for creating visually appealing plots like distribution plots, box plots, violin

Importing libraries

In [1]:

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

Read Dataset

In [2]:

```
# To read the .csv file
nf = pd.read_csv('C:\\Users\\91827\\Downloads\\Netflix userbase.csv')
nf
```

Out[2]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	Male	NaN	1
3	4	Standard	NaN	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	1
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	Male	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2500 rows × 10 columns

Applying Basic Functions

First 2 values

In [63]:

```
nf.head(2)
```

Out[63]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Pl Durati
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1 Mor
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1 Mor

Shape of the dataset

In [55]:

```
# checking the no. of rows and columns
nf.shape
```

Out[55]:

```
(2500, 10)
```

Print the name of columns

In [51]:

```
# List down all the column names
nf.columns
```

Out[51]:

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

Check for NULL Values

In [12]:

```
nf.isnull().sum()
```

Out[12]:

```
User ID              0
Subscription Type    1
Monthly Revenue      1
Join Date            0
Last Payment Date    0
Country              2
Age                  1
Gender               0
Device               1
Plan Duration        1
dtype: int64
```

Check unique values

In [49]:

```
nf.nunique()
```

Out[49]:

```
User ID              2493
Subscription Type       3
Monthly Revenue         6
Join Date             297
Last Payment Date      26
Country                10
Age                    26
Gender                  2
Device                  4
Plan Duration           1
dtype: int64
```

Check for Duplicate values

In [13]:

```
nf.duplicated().sum()
```

Out[13]:

```
0
```

Make a copy of the dataset

In [47]:

```
cd = nf.copy()
```

In [46]:

```
cd.shape
```

Out[46]:

```
(2500, 10)
```

Drop NULL values

In [57]:

```
nf=nf.dropna()
nf.shape
```

Out[57]:

```
(2493, 10)
```

Last 2 values

In [58]:

```
nf.tail(2)
```

Out[58]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Pla Duratio
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1 Mont
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1 Mont

checking datatype

In [59]:

```
nf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2493 entries, 0 to 2499
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   User ID            2493 non-null   int64  
 1   Subscription Type  2493 non-null   object 
 2   Monthly Revenue    2493 non-null   float64
 3   Join Date          2493 non-null   object 
 4   Last Payment Date  2493 non-null   object 
 5   Country            2493 non-null   object 
 6   Age                2493 non-null   float64
 7   Gender             2493 non-null   object 
 8   Device             2493 non-null   object 
 9   Plan Duration      2493 non-null   object 
dtypes: float64(2), int64(1), object(7)
memory usage: 214.2+ KB
```

In [14]:

```
# To count the number of females and males?
nf['Gender'].value_counts()
```

Out[14]:

```
Basic       998
Standard    768
Premium     733
Name: Subscription Type, dtype: int64
```

In [ ]:

```
# To print only tablet device users
nf[nf["Device"] == "Tablet"]
print(nf[nf["Device"] == "Tablet"])
```

Data cleaning:

- 1.isnull() : To check the data having null values or not.
- 2.notnull(): To get the not null values.
- 3.dropna(): To remove the null values.
- 4.fillna(): To fill the null values with meaningfull data.
- 5.replace(): To replace the data
- 6.sort_values(): To sort the data
- NaN : Not a number.

In [65]:

```
# 1.To count the number of null values
nf.isnull().sum()
```

Out[65]:

```
User ID              0
Subscription Type    1
Monthly Revenue      1
Join Date            0
Last Payment Date    0
Country              2
Age                  1
Gender               0
Device               1
Plan Duration        1
dtype: int64
```

In [67]:

```
# 1.2 To print the only null values records
nf[nf.isnull().any(axis=1)]
```

Out[67]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	F Dura
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	Male	NaN	1 M
3	4	Standard	NaN	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	1 M
5	6	Premium	15.0	18-03-2022	27-06-2023	France	NaN	Female	Smart TV	1 M
7	8	NaN	10.0	02-04-2023	24-06-2023	Mexico	39.0	Female	Laptop	1 M
9	10	Premium	15.0	07-01-2023	22-06-2023	NaN	44.0	Female	Smart TV	1 M
12	13	Standard	12.0	30-11-2021	27-06-2023	United Kingdom	48.0	Female	Laptop	I
13	14	Basic	10.0	01-08-2022	26-06-2023	NaN	27.0	Male	Smartphone	1 M

In [7]:

```
# 2.To count the not null records?
nf.notnull().sum()
```

Out[7]:

```
User ID              2500
Subscription Type    2499
Monthly Revenue      2499
Join Date            2500
Last Payment Date    2500
Country              2498
Age                  2499
Gender               2500
Device               2499
Plan Duration        2499
dtype: int64
```

In [9]:

```
# 2.2 To print the not null records
nf[nf.notnull().all(axis=1)]
```

Out[9]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
6	7	Standard	12.0	09-12-2021	25-06-2023	Brazil	46.0	Male	Tablet	1
8	9	Standard	12.0	20-10-2022	23-06-2023	Spain	37.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	Male	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2493 rows × 10 columns

In [89]:

```
# 3.dropna(): To remove the null values..
nf.dropna()
```

Out[89]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
6	7	Standard	12.0	09-12-2021	25-06-2023	Brazil	46.0	Male	Tablet	1
8	9	Standard	12.0	20-10-2022	23-06-2023	Spain	37.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	Male	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2493 rows × 10 columns

In [14]:

```
# 4.fillna(): To fill the null values with meaningfull data.
nf.fillna('missing value',inplace=True)
nf
```

Out[14]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	Male	missing value	1
3	4	Standard	missing value	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	1
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	Male	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2500 rows × 10 columns

In [125]:

```
dc = pd.read_csv('Netflix userbase.csv')
dc
```

Out[125]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	Male	NaN	1
3	4	Standard	NaN	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	Male	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2500 rows × 10 columns

In [126]:

```
# 4.2 To fill the specific column name null values
dc['Device'].fillna('Computer',inplace=True)
dc
```

Out[126]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	Male	Computer	1
3	4	Standard	NaN	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	Male	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2500 rows × 10 columns

In [127]:

```
#  5. replace(): To replace the data
dc['Gender'].replace(to_replace='Male',value='M',inplace=True)
dc
```

Out[127]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	M	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	M	Computer	1
3	4	Standard	NaN	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	M	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	M	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2500 rows × 10 columns

In [134]:

```
# 6.Sort_values(): Sorting by column 'Country'
nf.sort_values(by=['Country'])
```

Out[134]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
273	274	Premium	10.0	01-08-2022	24-06-2023	Australia	27.0	Female	Tablet	1
988	989	Basic	12.0	08-11-2022	01-07-2023	Australia	34.0	Male	Smartphone	1
2143	2144	Standard	14.0	15-07-2022	10-07-2023	Australia	47.0	Female	Smartphone	1
1258	1259	Premium	13.0	11-10-2022	03-07-2023	Australia	41.0	Female	Laptop	1
413	414	Premium	14.0	21-08-2022	27-06-2023	Australia	47.0	Female	Laptop	1
...	...	...	...	...	...	...	...	...	...	...
1524	1525	Basic	14.0	24-08-2022	04-07-2023	United States	50.0	Female	Tablet	1
1479	1480	Basic	13.0	03-11-2022	07-07-2023	United States	49.0	Female	Laptop	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1
13	14	Basic	10.0	01-08-2022	26-06-2023	missing value	27.0	Male	Smartphone	1
9	10	Premium	15.0	07-01-2023	22-06-2023	missing value	44.0	Female	Smart TV	1

2500 rows × 10 columns

Data Filtering

In [150]:

```
df= pd.read_csv('Netflix userbase.csv')
df
```

Out[150]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	Male	NaN	1
3	4	Standard	NaN	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	Male	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2500 rows × 10 columns

In [158]:

```
# 1.Select particular columns using column values in a dataframe method
df[["Monthly Revenue","Plan Duration"]]
```

Out[158]:

	Monthly Revenue	Plan Duration
0	10.0	1 Month
1	15.0	1 Month
2	12.0	1 Month
3	NaN	NaN
4	10.0	1 Month
...	...	...
2495	14.0	1 Month
2496	15.0	1 Month
2497	12.0	1 Month
2498	13.0	1 Month
2499	15.0	1 Month

2500 rows × 2 columns

In [159]:

```
# 2.Slicing using index to filter rows
df[1:3]
```

Out[159]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Plan Duration
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1 Month
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	Male	NaN	1 Month

In [160]:

```
# 3.Filter rows and columns using iloc() function
# iloc() method selects rows and columns based on the index/position values.
df.iloc[ 0:3 , 0:2]
```

Out[160]:

	User ID	Subscription Type
0	1	Basic
1	2	Premium
2	3	Standard

In [161]:

```
# 4.Filter rows and columns using loc() method(slicing using labels)
df.loc[ [3,4] , ['Country','Device'] ]
```

Out[161]:

	Country	Device
3	Australia	Laptop
4	Germany	Smartphone

In [162]:

```
# 5.Filter Devices by Tablet
df[nf['Device']=='Tablet']
```

Out[162]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Pla Duratic
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1 Mon
6	7	Standard	12.0	09-12-2021	25-06-2023	Brazil	46.0	Male	Tablet	1 Mon
11	12	Premium	15.0	23-03-2023	28-06-2023	Canada	45.0	Male	Tablet	1 Mon
15	16	Premium	15.0	07-04-2022	27-06-2023	France	36.0	NaN	Tablet	1 Mon
19	20	Basic	10.0	27-05-2023	22-06-2023	Italy	41.0	Male	Tablet	1 Mon
...	...	...	...	...	...	...	...	...	...	
2474	2475	Basic	12.0	05-09-2022	11-07-2023	Germany	37.0	Female	Tablet	1 Mon
2475	2476	Basic	13.0	31-08-2022	15-07-2023	France	28.0	Male	Tablet	1 Mon
2477	2478	Standard	10.0	16-08-2022	13-07-2023	Mexico	47.0	Male	Tablet	1 Mon
2484	2485	Premium	14.0	25-07-2022	12-07-2023	United States	47.0	Female	Tablet	1 Mon
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1 Mon

633 rows × 10 columns

In [163]:

```
# 6.Filter single item from the dataset
df.iat[1,1]
```

Out[163]:

```
'Premium'
```

In [164]:

```
df.at[1, 'Country']
```

Out[164]:

```
'Canada'
```

In [165]:

```
# 7.Using regular expression to extract all columns which has letter 'a' or 'A' in its n
df.filter(regex ='[aA]')
```

Out[165]:

	Join Date	Last Payment Date	Age	Plan Duration
0	15-01-2022	10-06-2023	28.0	1 Month
1	05-09-2021	22-06-2023	35.0	1 Month
2	28-02-2023	27-06-2023	42.0	1 Month
3	10-07-2022	26-06-2023	51.0	NaN
4	01-05-2023	28-06-2023	33.0	1 Month
...	...	...	...	...
2495	25-07-2022	12-07-2023	28.0	1 Month
2496	04-08-2022	14-07-2023	33.0	1 Month
2497	09-08-2022	15-07-2023	38.0	1 Month
2498	12-08-2022	12-07-2023	48.0	1 Month
2499	13-08-2022	12-07-2023	35.0	1 Month

2500 rows × 4 columns

In [166]:

```
# 8.count various Subscription Types
df['Subscription Type'].value_counts()
```

Out[166]:

```
Basic       998
Standard    768
Premium     733
Name: Subscription Type, dtype: int64
```

In [169]:

```
# 9.TO print the least Monthly Revenue records from a df?
df[df['Monthly Revenue']==df['Monthly Revenue'].min()]
```

Out[169]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
7	8	NaN	10.0	02-04-2023	24-06-2023	Mexico	39.0	Female	Laptop	1
10	11	Basic	10.0	16-05-2022	22-06-2023	United States	31.0	Female	Smartphone	1
13	14	Basic	10.0	01-08-2022	26-06-2023	NaN	27.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2473	2474	Premium	10.0	06-09-2022	12-07-2023	Australia	49.0	Male	Smart TV	1
2477	2478	Standard	10.0	16-08-2022	13-07-2023	Mexico	47.0	Male	Tablet	1
2480	2481	Premium	10.0	05-08-2022	12-07-2023	Spain	33.0	Female	Laptop	1
2485	2486	Basic	10.0	25-07-2022	14-07-2023	United States	40.0	Female	Smartphone	1
2489	2490	Standard	10.0	17-07-2022	15-07-2023	Germany	35.0	Male	Smart TV	1

409 rows × 10 columns

In [187]:

```
# 10. To print the Country name starting with 'F' from dataset?
nf[nf['Country'].str.startswith('F')] # taking nf dataset becoz of nan values in df
```

Out[187]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device
5	6	Premium	15.0	18-03-2022	27-06-2023	France	missing value	Female	Smart TV
15	16	Premium	15.0	07-04-2022	27-06-2023	France	36.0	Male	Tablet
25	26	Premium	15.0	12-01-2022	27-06-2023	France	29.0	Male	Smartphone
35	36	Premium	15.0	01-03-2022	27-06-2023	France	35.0	Male	Tablet
45	46	Premium	15.0	23-02-2022	27-06-2023	France	34.0	Male	Smartphone
...	...	...	...	...	...	...	...	...	...
2430	2431	Premium	12.0	15-08-2022	14-07-2023	France	29.0	Female	Smart TV
2445	2446	Premium	11.0	18-10-2022	12-07-2023	France	46.0	Male	Tablet
2460	2461	Premium	13.0	13-11-2022	12-07-2023	France	41.0	Male	Laptop
2475	2476	Basic	13.0	31-08-2022	15-07-2023	France	28.0	Male	Tablet
2490	2491	Premium	13.0	18-07-2022	11-07-2023	France	41.0	Female	Smartphone

183 rows × 10 columns

Grouping

In [10]:

```
gp= pd.read_csv('Netflix userbase.csv')
gp
```

Out[10]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	Male	NaN	1
3	4	Standard	NaN	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	Male	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2500 rows × 10 columns

In [11]:

```
# 1.applying groupby() function to group the data on Device value.
gp = nf.groupby('Device')
gp.first()
```

Out[11]:

Device	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Plan Duration
Laptop	4	Standard	10.0	10-07-2022	26-06-2023	Australia	51.0	Female	1 Month
Smart TV	6	Premium	15.0	18-03-2022	27-06-2023	France	44.0	Female	1 Month
Smartphone	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	1 Month
Tablet	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	1 Month

In [12]:

```
# 2.We use the function get_group() to find the entries contained in any of the groups.
gp.get_group('Smartphone')
```

Out[12]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
8	9	Standard	12.0	20-10-2022	23-06-2023	Spain	37.0	Male	Smartphone	1
10	11	Basic	10.0	16-05-2022	22-06-2023	United States	31.0	Female	Smartphone	1
13	14	Basic	10.0	01-08-2022	26-06-2023	NaN	27.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2470	2471	Basic	10.0	17-10-2022	11-07-2023	United States	46.0	Male	Smartphone	1
2485	2486	Basic	10.0	25-07-2022	14-07-2023	United States	40.0	Female	Smartphone	1
2487	2488	Standard	11.0	18-07-2022	13-07-2023	United Kingdom	29.0	Female	Smartphone	1
2488	2489	Basic	11.0	17-07-2022	12-07-2023	Australia	48.0	Female	Smartphone	1
2490	2491	Premium	13.0	18-07-2022	11-07-2023	France	41.0	Female	Smartphone	1

621 rows × 10 columns

Sorting

In [19]:

```
sg= pd.read_csv('Netflix userbase.csv')
sg
```

Out[19]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	Male	NaN	1
3	4	Standard	NaN	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	Male	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2500 rows × 10 columns

In [24]:

```
# 1.Sorting by column "Age"
sg.sort_values(by=['Age'], ascending=True)
```

Out[24]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
20	21	Premium	15.0	10-06-2023	22-06-2023	United States	26.0	Female	Laptop	1
1297	1298	Standard	14.0	26-06-2022	03-07-2023	United States	27.0	Male	Laptop	1
1045	1046	Basic	13.0	06-10-2022	03-07-2023	United States	27.0	Male	Laptop	1
1077	1078	Standard	14.0	17-07-2022	02-07-2023	United Kingdom	27.0	Male	Laptop	1
1081	1082	Basic	13.0	28-06-2022	02-07-2023	Brazil	27.0	Male	Tablet	1
...	...	...	...	...	...	...	...	...	...	...
849	850	Basic	13.0	13-11-2022	30-06-2023	United States	51.0	Female	Smart TV	1
1110	1111	Premium	13.0	10-09-2022	01-07-2023	France	51.0	Male	Smart TV	1
1758	1759	Standard	10.0	06-11-2022	07-07-2023	Spain	51.0	Male	Smart TV	1
1119	1120	Basic	13.0	10-10-2022	04-07-2023	United States	51.0	Female	Smartphone	1
5	6	Premium	15.0	18-03-2022	27-06-2023	France	NaN	Female	Smart TV	1

2500 rows × 10 columns

In [25]:

```
# 2.Sorting Pandas Data frame by putting missing values first
sg.sort_values(by=['Age'], na_position='first')
```

Out[25]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
5	6	Premium	15.0	18-03-2022	27-06-2023	France	NaN	Female	Smart TV	1
20	21	Premium	15.0	10-06-2023	22-06-2023	United States	26.0	Female	Laptop	1
1297	1298	Standard	14.0	26-06-2022	03-07-2023	United States	27.0	Male	Laptop	1
1045	1046	Basic	13.0	06-10-2022	03-07-2023	United States	27.0	Male	Laptop	1
1077	1078	Standard	14.0	17-07-2022	02-07-2023	United Kingdom	27.0	Male	Laptop	1
...	...	...	...	...	...	...	...	...	...	
215	216	Premium	14.0	08-10-2022	24-06-2023	France	51.0	Female	Smart TV	1
849	850	Basic	13.0	13-11-2022	30-06-2023	United States	51.0	Female	Smart TV	1
1110	1111	Premium	13.0	10-09-2022	01-07-2023	France	51.0	Male	Smart TV	1
1758	1759	Standard	10.0	06-11-2022	07-07-2023	Spain	51.0	Male	Smart TV	1
1119	1120	Basic	13.0	10-10-2022	04-07-2023	United States	51.0	Female	Smartphone	1

2500 rows × 10 columns

In [26]:

```
# 3.Sorting Data frames by multiple columns
# Sorting by columns "Country" and then "Director"
sg.sort_values(by=['Monthly Revenue', 'Age'])
```

Out[26]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
13	14	Basic	10.0	01-08-2022	26-06-2023	NaN	27.0	Male	Smartphone	1
70	71	Basic	10.0	26-01-2022	20-06-2023	United States	27.0	Male	Smartphone	1
259	260	Basic	10.0	17-07-2022	26-06-2023	Italy	27.0	Female	Smartphone	1
273	274	Premium	10.0	01-08-2022	24-06-2023	Australia	27.0	Female	Tablet	1
298	299	Premium	10.0	30-10-2022	27-06-2023	Spain	27.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	...
2039	2040	Standard	15.0	08-11-2022	08-07-2023	Germany	51.0	Male	Laptop	1
2227	2228	Standard	15.0	16-09-2022	10-07-2023	United States	51.0	Male	Tablet	1
2425	2426	Premium	15.0	23-07-2022	12-07-2023	United States	51.0	Male	Smartphone	1
5	6	Premium	15.0	18-03-2022	27-06-2023	France	NaN	Female	Smart TV	1
3	4	Standard	NaN	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	

2500 rows × 10 columns

Aggregations

- To work on numerical data

In [6]:

```
ag= pd.read_csv('Netflix userbase.csv')
ag
```

Out[6]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Du
0	1	Basic	10.0	15-01-2022	10-06-2023	United States	28.0	Male	Smartphone	1
1	2	Premium	15.0	05-09-2021	22-06-2023	Canada	35.0	Female	Tablet	1
2	3	Standard	12.0	28-02-2023	27-06-2023	United Kingdom	42.0	Male	NaN	1
3	4	Standard	NaN	10-07-2022	26-06-2023	Australia	51.0	Female	Laptop	
4	5	Basic	10.0	01-05-2023	28-06-2023	Germany	33.0	Male	Smartphone	1
...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14.0	25-07-2022	12-07-2023	Spain	28.0	Female	Smart TV	1
2496	2497	Basic	15.0	04-08-2022	14-07-2023	Spain	33.0	Female	Smart TV	1
2497	2498	Standard	12.0	09-08-2022	15-07-2023	United States	38.0	Male	Laptop	1
2498	2499	Standard	13.0	12-08-2022	12-07-2023	Canada	48.0	Female	Tablet	1
2499	2500	Basic	15.0	13-08-2022	12-07-2023	United States	35.0	Female	Smart TV	1

2500 rows × 10 columns

In [28]:

```
# 1.min
ag.min()
```

```
C:\Users\91827\AppData\Local\Temp\ipykernel_16172\1429976717.py:1: Future
Warning: Dropping of nuisance columns in DataFrame reductions (with 'nume
ric_only=None') is deprecated; in a future version this will raise TypeEr
ror.  Select only valid columns before calling the reduction.
  ag.min()
```

Out[28]:

```
User ID                     1
Monthly Revenue          10.0
Join Date            01-03-2022
Last Payment Date    01-07-2023
Age                      26.0
dtype: object
```

In [29]:

```
# 2.max
ag.max()
```

```
C:\Users\91827\AppData\Local\Temp\ipykernel_16172\2130797385.py:1: Future
Warning: Dropping of nuisance columns in DataFrame reductions (with 'nume
ric_only=None') is deprecated; in a future version this will raise TypeEr
ror.  Select only valid columns before calling the reduction.
  ag.max()
```

Out[29]:

```
User ID                   2500
Monthly Revenue           15.0
Join Date            31-10-2022
Last Payment Date    30-06-2023
Age                      51.0
dtype: object
```

In [30]:

```
# 3.mean
ag.mean()
```

```
C:\Users\91827\AppData\Local\Temp\ipykernel_16172\2231397629.py:1: Future
Warning: Dropping of nuisance columns in DataFrame reductions (with 'nume
ric_only=None') is deprecated; in a future version this will raise TypeEr
ror.  Select only valid columns before calling the reduction.
  ag.mean()
```

Out[30]:

```
User ID            1250.500000
Monthly Revenue      12.508603
Age                  38.799520
dtype: float64
```

In [32]:

```
# 4.std
ag.std()
```

```
C:\Users\91827\AppData\Local\Temp\ipykernel_16172\1087247426.py:1: Future
Warning: Dropping of nuisance columns in DataFrame reductions (with 'nume
ric_only=None') is deprecated; in a future version this will raise TypeEr
ror.  Select only valid columns before calling the reduction.
  ag.std()
```

Out[32]:

```
User ID            721.832160
Monthly Revenue      1.687158
Age                  7.170534
dtype: float64
```

In [33]:

```
# 5.var
ag.var()
```

```
C:\Users\91827\AppData\Local\Temp\ipykernel_16172\3862806996.py:1: Future
Warning: Dropping of nuisance columns in DataFrame reductions (with 'nume
ric_only=None') is deprecated; in a future version this will raise TypeEr
ror.  Select only valid columns before calling the reduction.
  ag.var()
```

Out[33]:

```
User ID            521041.666667
Monthly Revenue         2.846503
Age                    51.416557
dtype: float64
```

In [7]:

```
# Statistics of data set
ag.describe().T.style.background_gradient(cmap='turbo')
```

Out[7]:

	count	mean	std	min	25%	50%	
User ID	2500.000000	1250.500000	721.832160	1.000000	625.750000	1250.500000	1875.25
Monthly Revenue	2499.000000	12.508603	1.687158	10.000000	11.000000	12.000000	14.0(
Age	2499.000000	38.799520	7.170534	26.000000	32.000000	39.000000	45.0(

Data Visualization

installing matplotlib

In [42]:

```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\91827\anaconda3\lib
\site-packages (3.5.1)
Requirement already satisfied: cycler>=0.10 in c:\users\91827\anaconda3\l
ib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\91827\anacon
da3\lib\site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\91827\ana
conda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\91827\anacond
a3\lib\site-packages (from matplotlib) (3.0.4)
Requirement already satisfied: numpy>=1.17 in c:\users\91827\anaconda3\li
b\site-packages (from matplotlib) (1.21.5)
Requirement already satisfied: pillow>=6.2.0 in c:\users\91827\anaconda3
\lib\site-packages (from matplotlib) (9.0.1)
Requirement already satisfied: packaging>=20.0 in c:\users\91827\anaconda
3\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\91827\anacon
da3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: six>=1.5 in c:\users\91827\anaconda3\lib\s
ite-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

importing matplotlib

In [43]:

```
import matplotlib as mpl
```

In [2]:

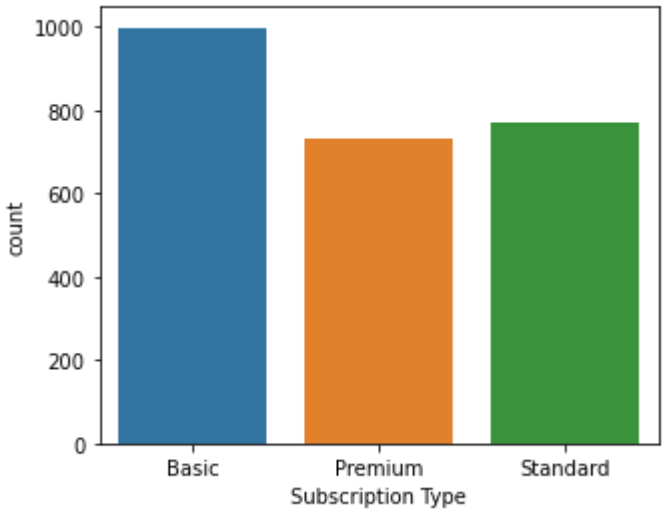
```
import matplotlib.pyplot as plt
```

1.Subscription Type

Which subscription plan are most people using ?

In [6]:

```
plt.figure(figsize=(5,4))
sns.countplot(data=nf,x='Subscription Type')
plt.show()
```

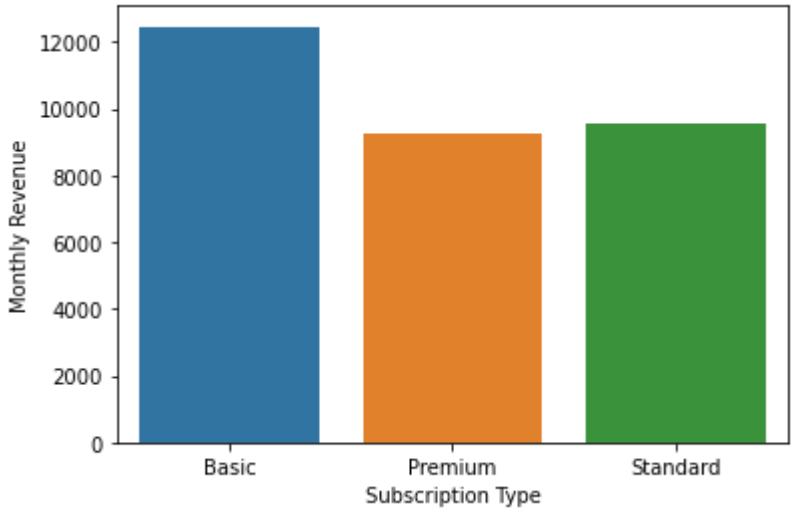


- Basic subscription plan are most people using

Which subscription type generate the most revenue ?

In [12]:

```
sm = nf.groupby('Subscription Type').sum().reset_index()
sns.barplot(data=sm,x='Subscription Type', y='Monthly Revenue')
plt.show()
```

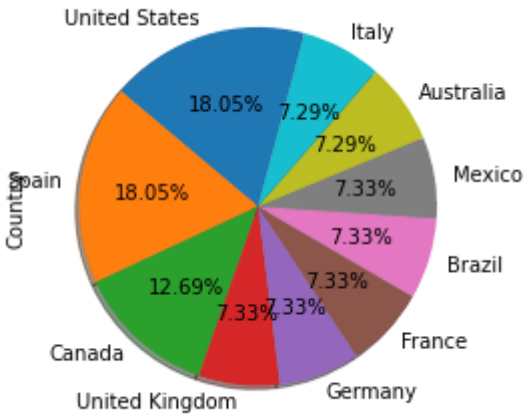


- Basic plan generates maximum revenue

Country

In [8]:

```
nf.Country.value_counts().plot(kind='pie',shadow='True' ,autopct='%1.2f%%', startangle =
plt.show()
```

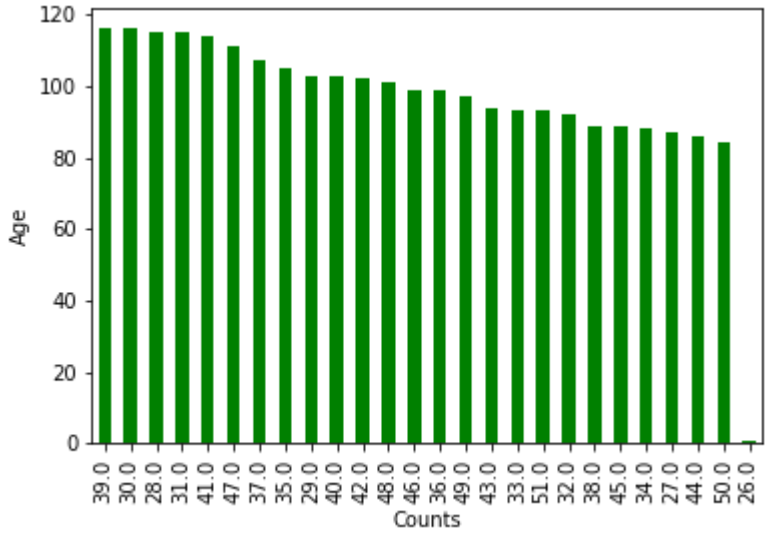


- Spain and United States have more customers

Age

In [12]:

```
nf.Age.value_counts().plot(kind='bar',color='g',figsize=(6,4))
plt.ylabel("Age")
plt.xlabel("Counts")
plt.show()
```

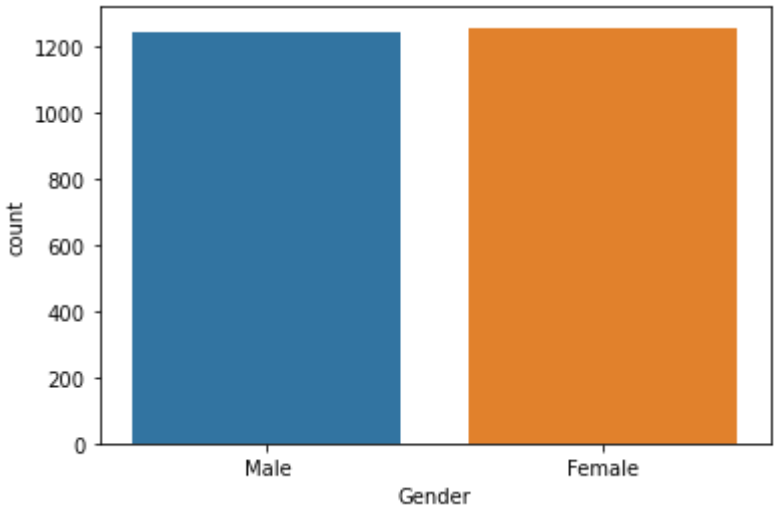


- Maximum users are aged 39 followed by 30,28,31,41

Gender

In [22]:

```
sns.countplot(data=nf,x='Gender')
plt.show()
```

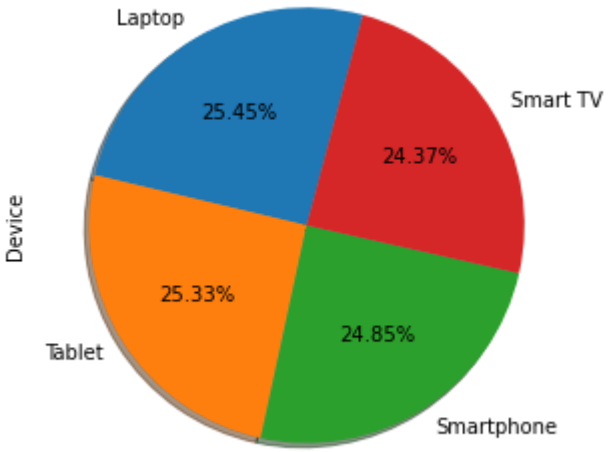


- Male & Female has equal counts

Device

In [13]:

```
nf.Device.value_counts().plot(kind='pie',shadow='True' ,autopct='%1.2f%%', startangle =
plt.show()
```

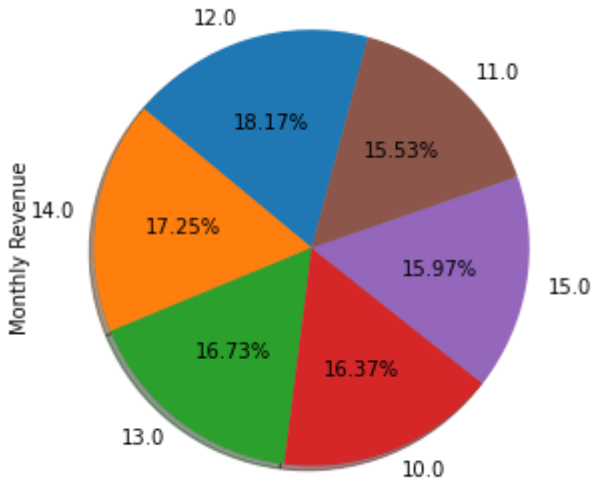


- MY SUGGESTION

Monthly Revenue

In [14]:

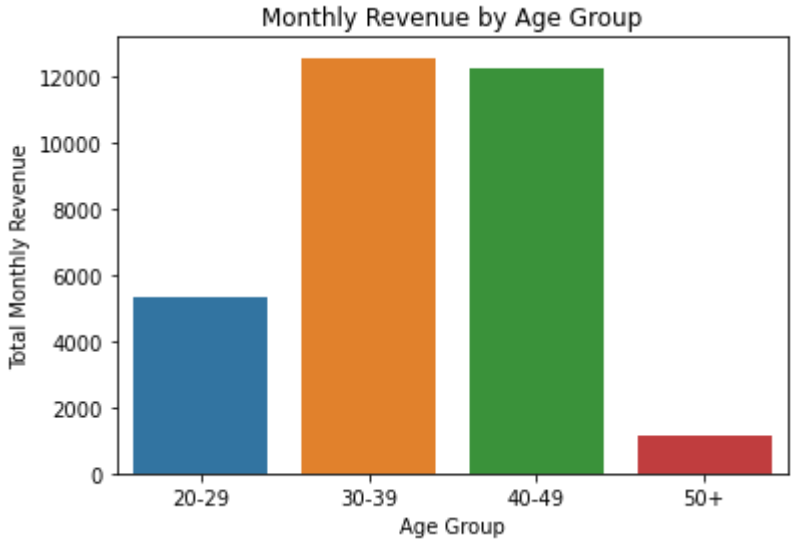
```
nf['Monthly Revenue'].value_counts().plot(kind='pie',shadow='True' ,autopct='%1.2f%%', s
plt.show()
```



Monthly Revenue by Age Group

In [15]:

```
age_bins = [20, 30, 40, 50, 60]
age_labels = ['20-29', '30-39', '40-49', '50+']
nf['age group'] = pd.cut(nf['Age'], bins=age_bins, labels=age_labels)
revenue_by_age = nf.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()
```

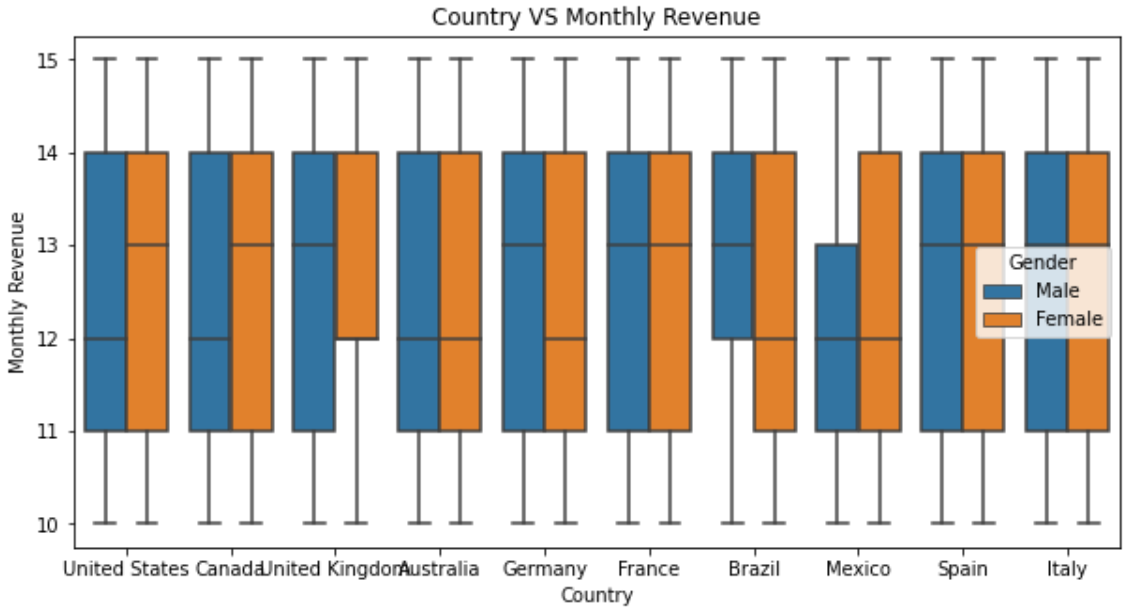


- Most of the users who are purchasing subscription has aged 30 to 49

Country VS Monthly Revenue

In [24]:

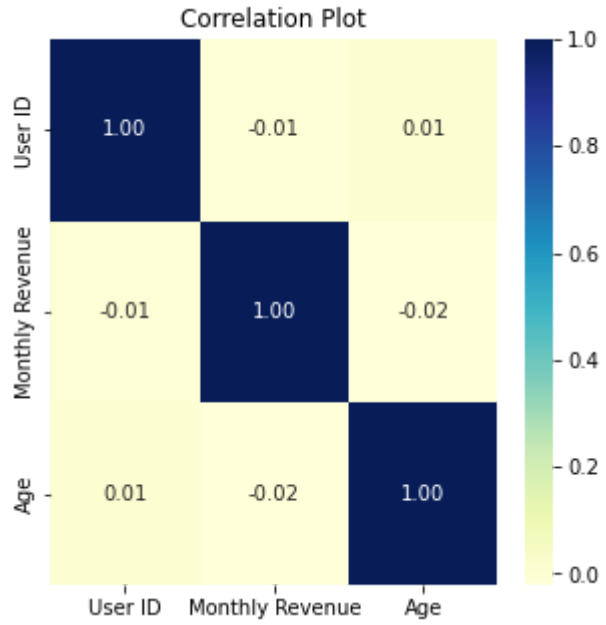
```
plt.figure(figsize=(10,5))
sns.boxplot(data=nf,x='Country', y='Monthly Revenue',hue='Gender',)
plt.title("Country VS Monthly Revenue")
plt.xlabel("Country")
plt.ylabel("Monthly Revenue")
plt.show()
```



Heatmap

In [17]:

```
corr_matrix = nf.corr()
plt.figure(figsize=(5, 5))
sns.heatmap(corr_matrix, annot=True, cmap="YlGnBu", fmt=".2f")
plt.title("Correlation Plot")
plt.show()
```



Final Insights

- Age group 20-29 spend more time on Netflix
- All countries have similar rate of spending time
- Age group 30-49 doing the most subscription and giving more revenue
- Basic type subscription has maximum purchasing

Future Scope

- The future scope could involve predictive modeling for churn prevention, content recommendation enhancements, and refining subscription plans based on user segments. Additionally, using AI to personalize user experiences and refining content delivery strategies could also be part of the future scope.

Advantages

- 1. Understanding User Preferences: By analyzing the subscription types chosen by users, Netflix can gain insights into which plans are most popular among their customer base. This knowledge can guide the platform in designing attractive subscription bundles and pricing models.
- 2. Revenue Trends and Growth: Monthly revenue analysis provides a clear view of Netflix's financial performance over time. Identifying revenue trends can help the company track its growth and

assess the effectiveness of marketing campaigns or promotional strategies.

- 3. Demographic Profiling: Analyzing user age, gender, and country data allows Netflix to better understand the diversity of its user base. Tailoring content to different demographics can increase engagement and retention.
- 4. Device Usage Insights: By examining the devices through which users access the platform, Netflix can optimize its app and website experiences for various platforms, leading to improved user satisfaction.
- 5. Plan Duration and User Retention: Calculating the average plan duration offers valuable insights into user retention. This information can help Netflix devise strategies to enhance customer loyalty and reduce churn rates.

Conclusion:

- In conclusion, this analysis and visualization of Netflix user data offer invaluable insights into user behavior, subscription trends, and revenue growth. By understanding user preferences, demographic profiles, and device usage patterns, Netflix can optimize its content offerings and user experience to better cater to its diverse customer base. The study also sheds light on user retention, helping Netflix focus on strategies to increase customer loyalty and reduce subscription cancellations. The data-driven approach presented in this analysis can serve as a foundation for Netflix's future business decisions, ensuring continuous improvement and innovation in the highly competitive streaming industry. As the digital landscape evolves, such analyses will remain crucial in maintaining Netflix's position as a market leader and providing its global audience with exceptional streaming experiences.

References websites

https://www.kaggle.com/ (https://www.kaggle.com/)

- Greeks for greeks

FINAL REVIEW QNS

In [23]:

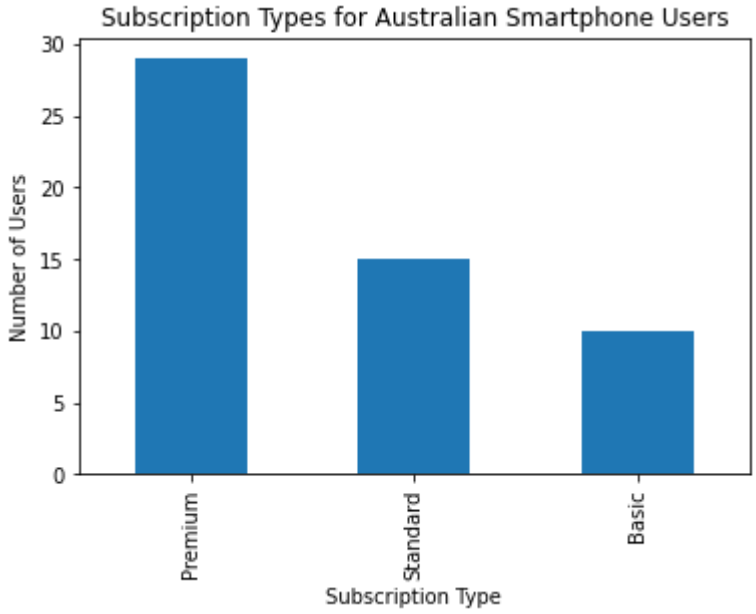
```
# Filter for Australian users with smartphone subscription
australian_smartphone_users = nf[(nf['Country'] == 'Australia') & (nf['Device'] == 'Smar
len(australian_smartphone_users)
```

Out[23]:

54

In [14]:

```
subscription_counts = australian_smartphone_users['Subscription Type'].value_counts()
subscription_counts.plot(kind='bar', title='Subscription Types for Australian Smartphone
plt.xlabel('Subscription Type')
plt.ylabel('Number of Users')
plt.show()
```



In [27]:

```
# Group data by country and calculate mean monthly revenue for each country
country_revenue = nf.groupby('Country')['Monthly Revenue'].mean()
country_revenue
```

Out[27]:

```
Country
Australia         12.425414
Brazil            12.486339
Canada            12.460568
France            12.606557
Germany           12.349727
Italy             12.648352
Mexico            12.224044
Spain             12.554324
United Kingdom    12.666667
United States     12.558758
Name: Monthly Revenue, dtype: float64
```

In [29]:

```
# Find the country with the highest mean monthly revenue
highest_revenue_country = country_revenue.idxmax()
highest_revenue_value = country_revenue.max()
print("Country with the highest mean monthly revenue:", highest_revenue_country)
print("Highest mean monthly revenue:", highest_revenue_value)
```

```
Country with the highest mean monthly revenue: United Kingdom
Highest mean monthly revenue: 12.666666666666666
```