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

1. Data Preprocessing:

 Load the provided dataset and perform initial data exploration.

In [2]:
```python
df = pd.read_excel(r'C:\Users\sutharsan\Downloads\customer_churn_large_dataset.xlsx')
```

In [3]:
```python
df.head()
```

Out[3]:

	CustomerID	Name	Age	Gender	Location	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB	Churn
0	1.0	Customer_1	63.0	Male	Los Angeles	17.0	73.36	236.0	0.0
1	2.0	Customer_2	62.0	Female	New York	1.0	48.76	172.0	0.0
2	3.0	Customer_3	24.0	Female	Los Angeles	5.0	85.47	460.0	0.0
3	4.0	Customer_4	36.0	Female	Miami	3.0	97.94	297.0	1.0
4	5.0	Customer_5	46.0	Female	Miami	19.0	58.14	266.0	0.0

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   CustomerID                  100000 non-null  float64
 1   Name                        100000 non-null  object 
 2   Age                         100000 non-null  float64
 3   Gender                      100000 non-null  object 
 4   Location                    100000 non-null  object 
 5   Subscription_Length_Months  100000 non-null  float64
 6   Monthly_Bill                100000 non-null  float64
 7   Total_Usage_GB              100000 non-null  float64
 8   Churn                       100000 non-null  float64
dtypes: float64(6), object(3)
memory usage: 5.7+ MB
```

In [5]:
```python
# df.shape
```

Handle missing data and outliers

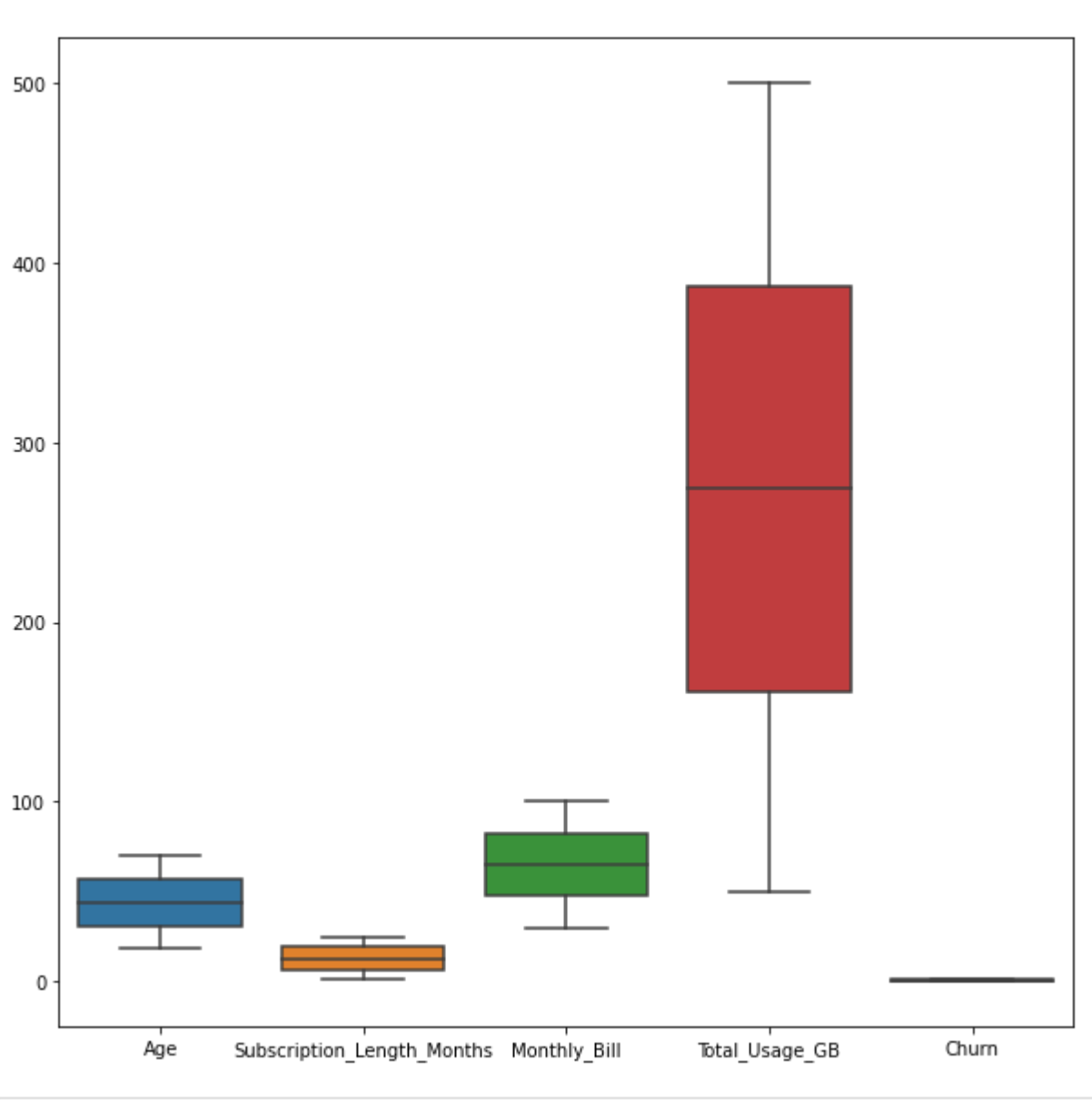
In [6]:
```python
df.isnull().sum()   # No Null Values
```

Out[6]:
```
CustomerID                    0
Name                          0
Age                           0
Gender                        0
Location                      0
Subscription_Length_Months    0
Monthly_Bill                  0
Total_Usage_GB                0
Churn                         0
dtype: int64
```

In [7]:
```python
import seaborn as sns
```

In [8]:
```python
plt.figure(figsize=(10,10))
df = df.drop('CustomerID',axis=1)    # no outliers
sns.boxplot(data = df)
```

Out[8]:
```
<AxesSubplot:>
```

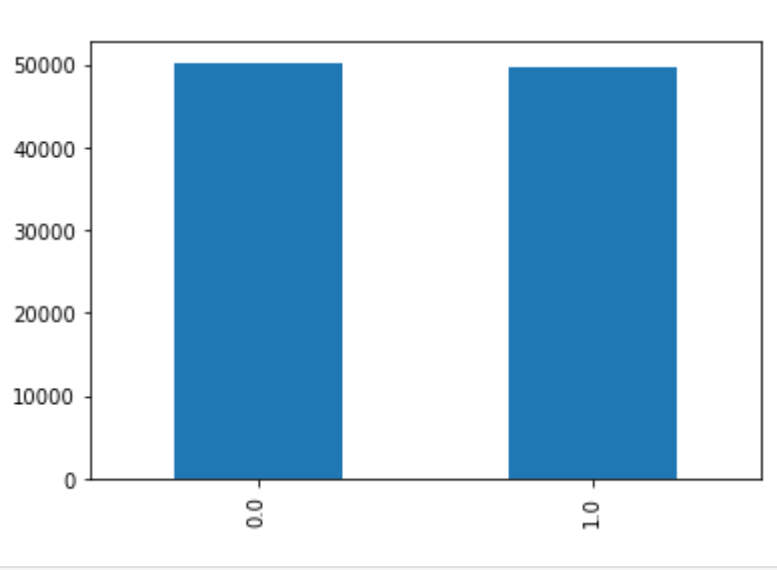


In [9]:
```python
df.columns
```

Out[9]:
```
Index(['Name', 'Age', 'Gender', 'Location', 'Subscription_Length_Months',
       'Monthly_Bill', 'Total_Usage_GB', 'Churn'],
      dtype='object')
```

In [10]:
```python
# as a classification model countplot for balanced data
df.Churn.value_counts().plot(kind="bar")  # balanced data
```

Out[10]:
```
<AxesSubplot:>
```



In [11]:
```python
df.dtypes
```

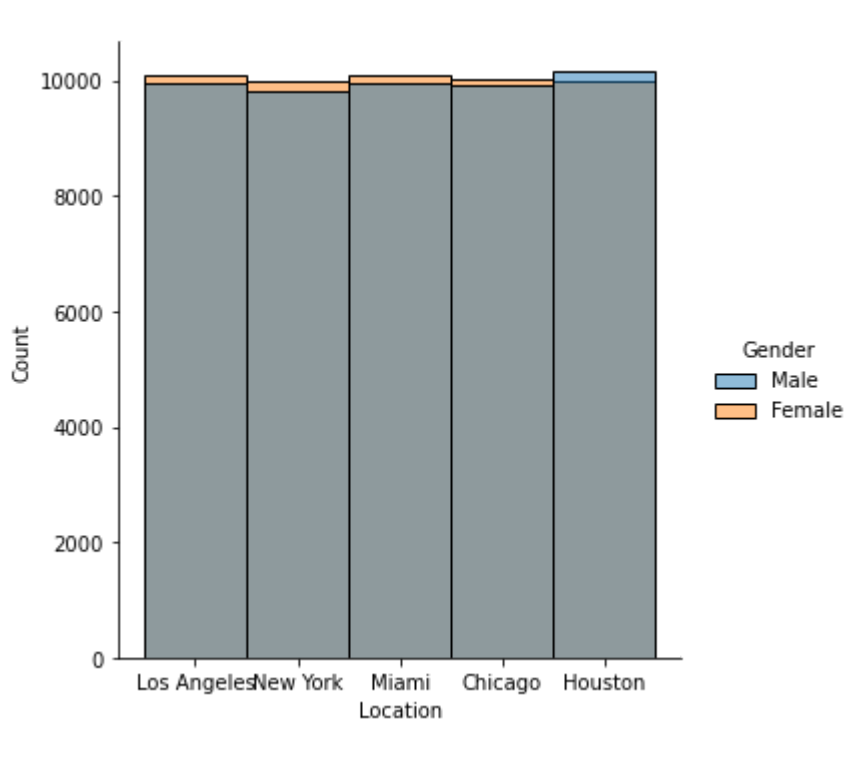
Out[11]:
```
Name                           object
Age                           float64
Gender                         object
Location                       object
Subscription_Length_Months    float64
Monthly_Bill                  float64
Total_Usage_GB                float64
Churn                         float64
dtype: object
```

In [12]:
```python
sns.countplot(x='Location',hue='Churn',data=df)
```

Out[12]:
```
<AxesSubplot:xlabel='Location', ylabel='count'>
```

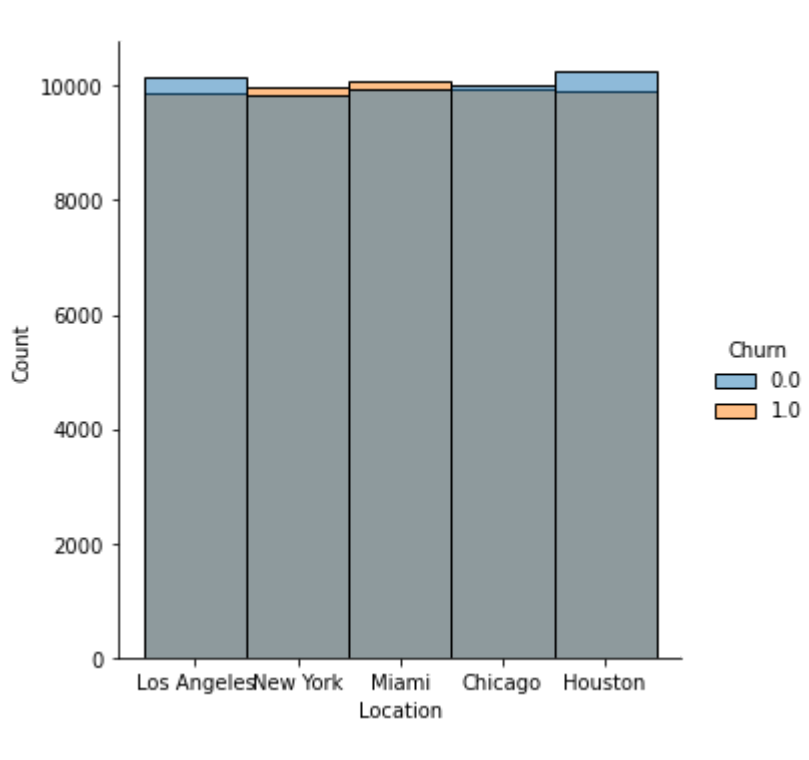
In [13]:
```python
sns.displot(x='Location',hue='Gender',data=df)   # In Houston Male customer are higher others Female are higher
```

Out[13]:
```
<seaborn.axisgrid.FacetGrid at 0x16d493b8>
```



In [14]:
```python
sns.displot(x='Location',hue='Churn',data=df)  # In NYC and Miami Location--> customer are churn
```

Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x16e3d928>
```



Feature Engineering -- onehotencoding

In [15]:
```python
df.Location.unique()
```

Out[15]:
```
array(['Los Angeles', 'New York', 'Miami', 'Chicago', 'Houston'],
      dtype=object)
```

In [16]:
```python
df = pd.get_dummies(df,columns=['Location'],drop_first= True)
```

In [17]:
```python
df.replace({'Male':0,'Female':1},inplace=True)
```

In [18]:
```python
df.head(2)
```

Out[18]:

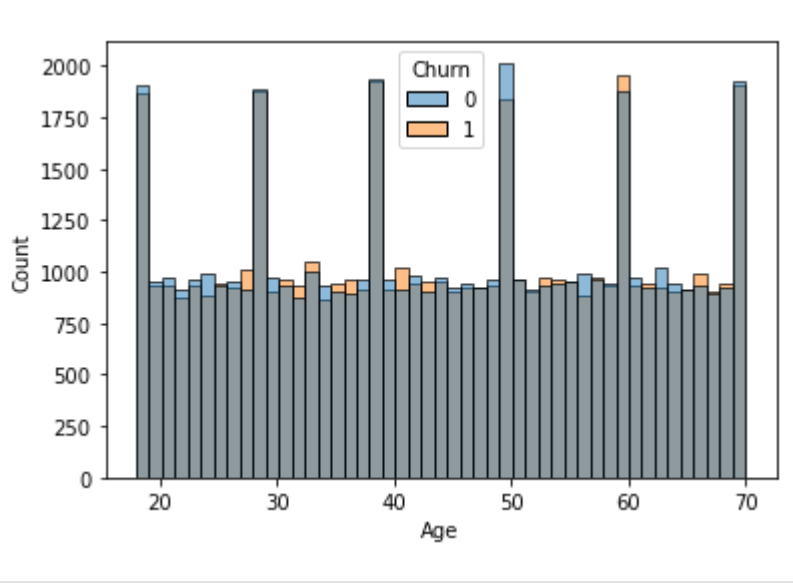
	Name	Age	Gender	Subscription_Length_Months	Monthly_Bill	Total_Usage_GB	Churn	Location_Houston	Location_Los Angeles	Location_Miami	Location_New York
0	Customer_1	63.0	0	17.0	73.36	236.0	0.0	0	1	0	0
1	Customer_2	62.0	1	1.0	48.76	172.0	0.0	0	0	0	1

In [19]:
```python
df['Churn'] = df['Churn'].astype(int)
```

Visualization

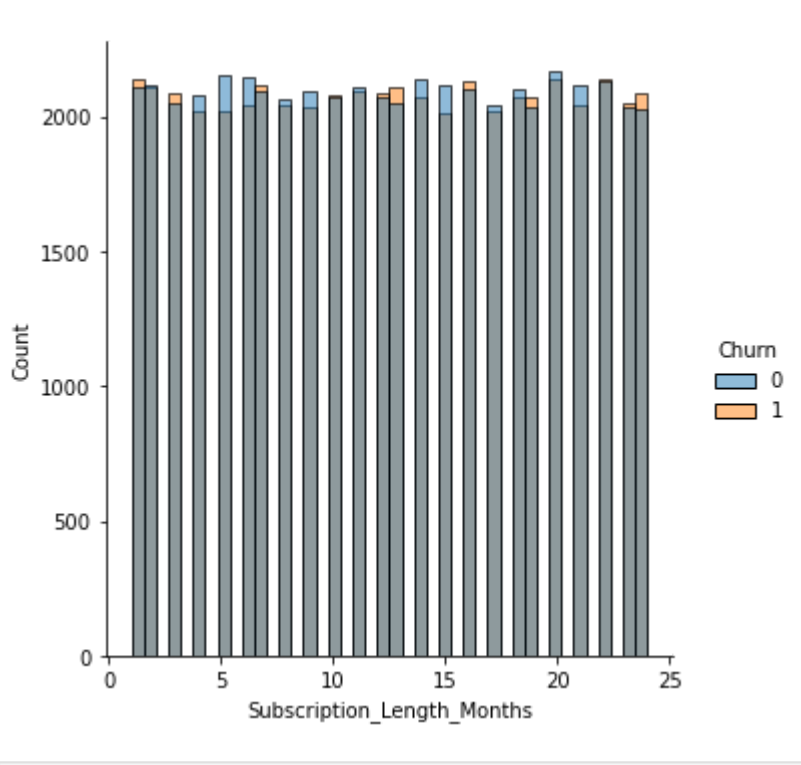
In [20]:
```python
sns.histplot(x='Age',hue='Churn',data=df) # Here age around 50 is high possible to churn
```

Out[20]:
```
<AxesSubplot:xlabel='Age', ylabel='Count'>
```



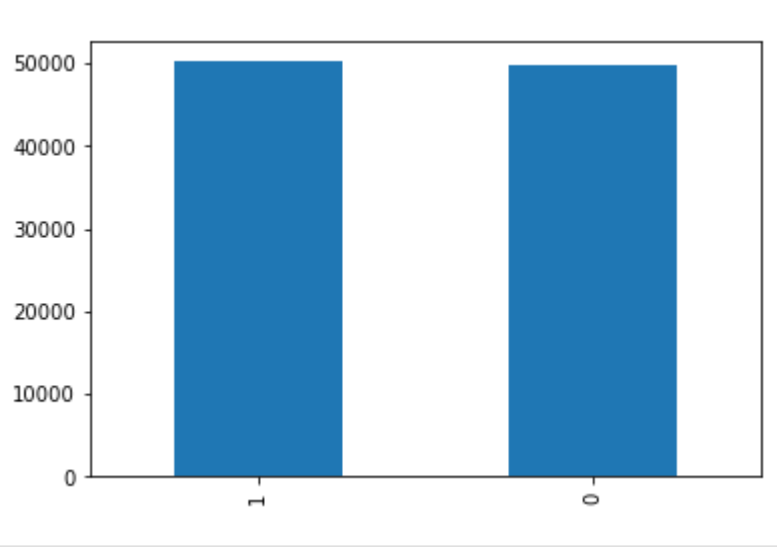
In [21]:
```python
sns.displot(x='Subscription_Length_Months',hue='Churn',data=df)
# distribution for subscription from this we can say by density is higher in 1,2,6,13,19,25 months are higher subscriber.
# like 2 month,5 month,10 month,20 month and 25 month subcriber are high compare to other months.
# we can see clearly which month subscriber are leaving.
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x16d49580>
```



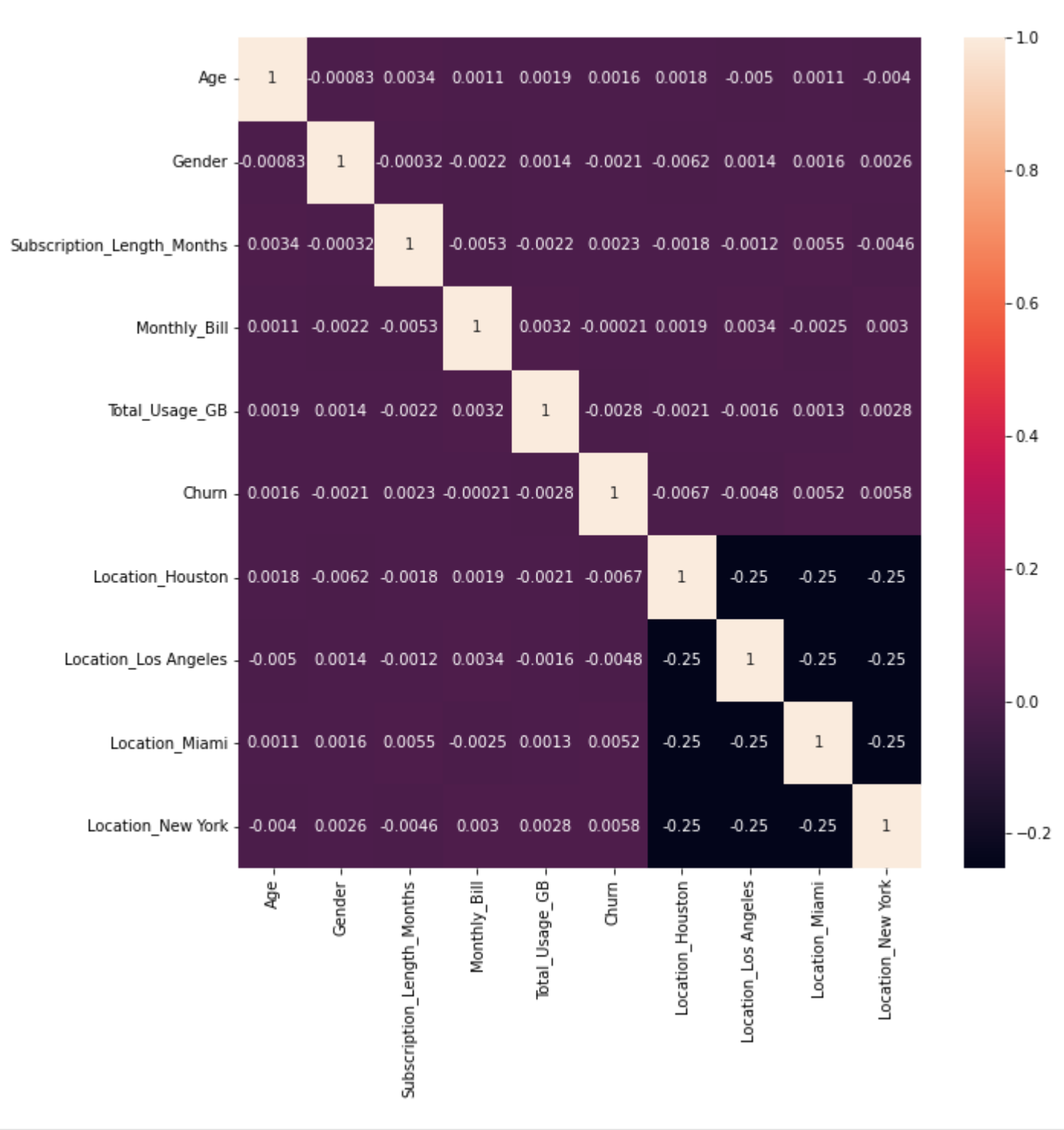
In [22]:
```python
df.Gender.value_counts().plot(kind="bar")     # we have equal no of gender subscriber.
```

Out[22]:
```
<AxesSubplot:>
```



In [23]:
```python
plt.figure(figsize=(10,10))
sns.heatmap(data = df.corr(),annot = True)
```

Out[23]:
```
<AxesSubplot:>
```



In [24]:
```python
y=df['Churn']
x=df.drop('Churn',axis=1,inplace=True)
```

In [25]:
```python
x=df.drop('Name',axis=1,inplace=True)
```

In [26]:
```python
y.isnull().sum()
```

Out[26]:
```
0
```

In [49]:
```python
x = df[['Age','Subscription_Length_Months']]
```

In [50]:
```python
df.columns
```

Out[50]:
```
Index(['Age', 'Gender', 'Subscription_Length_Months', 'Monthly_Bill',
       'Total_Usage_GB', 'Location_Houston', 'Location_Los Angeles',
       'Location_Miami', 'Location_New York'],
      dtype='object')
```

In [51]:
```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

In [52]:
```python
# Step 4 - Feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [53]:
```python
# Step 5 - Logistic regression classifier
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train, y_train)
```

Out[53]:
```
RandomForestClassifier()
```

In [54]:
```python
# Step 6 - Predicting logistic regression model on x_test
y_pred = rf.predict(x_test)
```

In [55]:
```python
# Step 7 - Confusion matrix
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)
accuracy = metrics.accuracy_score(y_test, y_pred)
print("Accuracy score:",accuracy)
precision = metrics.precision_score(y_test, y_pred)
print("Precision score:",precision)
recall = metrics.recall_score(y_test, y_pred)
print("Recall score:",recall)
```

```
[[7820 7205]
 [7691 7284]]
Accuracy score: 0.5034666666666666
Precision score: 0.5027262060873766
Recall score: 0.48641068447412356
```

In [56]:
```python
# Step 5 - Logistic regression classifier
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0, solver="liblinear")
classifier.fit(x_train, y_train)
```

Out[56]:
```
LogisticRegression(random_state=0, solver='liblinear')
```

In [57]:
```python
# Step 6 - Predicting logistic regression model on x_test
lr_pred = classifier.predict(x_test)
```

In [58]:
```python
# Step 7 - Confusion matrix
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, lr_pred)
print(cm)
accuracy = metrics.accuracy_score(y_test, lr_pred)
print("Accuracy score:",accuracy)
precision = metrics.precision_score(y_test, lr_pred)
print("Precision score:",precision)
recall = metrics.recall_score(y_test, lr_pred)
print("Recall score:",recall)
```

```
[[15025     0]
 [14975     0]]
Accuracy score: 0.5008333333333334
Precision score: 0.0
Recall score: 0.0
C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [59]:
```python
# Accuracy is low so HyperParameter tuning
from sklearn.model_selection import GridSearchCV
param_grid = [
    {'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
    'C' : np.logspace(-4, 4, 20),
    'solver' : ['lbfgs','newton-cg','liblinear','sag']
    }
]
```

In [60]:
```python
clf = GridSearchCV(classifier, param_grid = param_grid, cv = 3, verbose=True, n_jobs=-1)
```

In [61]:
```python
best_clf = clf.fit(x,y)
```

```
Fitting 3 folds for each of 320 candidates, totalling 960 fits
C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:372: FitFailedWarning: 
480 fits failed out of a total of 960.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver newton-cg supports only 'l2' or 'none' penalties, got l1 penalty.

--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver sag supports only 'l2' or 'none' penalties, got l1 penalty.

--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got elasticnet penalty.

--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver newton-cg supports only 'l2' or 'none' penalties, got elasticnet penalty.

--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 457, in _check_solver
    raise ValueError(
ValueError: Only 'saga' solver supports elasticnet penalty, got solver=liblinear.

--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver sag supports only 'l2' or 'none' penalties, got elasticnet penalty.

--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 464, in _check_solver
    raise ValueError("penalty='none' is not supported for the liblinear solver")
ValueError: penalty='none' is not supported for the liblinear solver

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\sutharsan\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:969: UserWarning: One or more of the test scores are non-finite: [       nan
nan 0.50221           nan 0.50065003 0.50065003
 0.50208    0.50065003        nan        nan        nan        nan
 0.50064003 0.50060003        nan 0.50060003        nan        nan
 0.50221           nan 0.50064003 0.50064003 0.50216    0.50064003
        nan        nan        nan        nan 0.50064003 0.50060003
        nan 0.50060003        nan        nan 0.50221           nan
 0.50064003 0.50065003 0.50172001 0.50064003        nan        nan
        nan        nan 0.50064003 0.50060003        nan 0.50060003
        nan        nan 0.50221           nan 0.50064003 0.50060003
 0.50114002 0.50060003        nan        nan        nan        nan
 0.50064003 0.50060003        nan 0.50060003        nan        nan
 0.50221           nan 0.50064003 0.50060003 0.50061003 0.50060003
        nan        nan        nan        nan 0.50064003 0.50060003
        nan 0.50060003        nan        nan 0.50221           nan
 0.50064003 0.50060003 0.50091003 0.50060003        nan        nan
        nan        nan 0.50064003 0.50060003        nan 0.50060003
        nan        nan 0.50181001        nan 0.50064003 0.50060003
 0.50066003 0.50060003        nan        nan        nan        nan
 0.50064003 0.50060003        nan 0.50060003        nan        nan
 0.50062003        nan 0.50064003 0.50060003 0.50064003 0.50060003
        nan        nan        nan        nan 0.50064003 0.50060003
        nan 0.50060003        nan        nan 0.50095003        nan
 0.50064003 0.50060003 0.50064003 0.50060003        nan        nan
        nan        nan 0.50064003 0.50060003        nan 0.50060003
        nan        nan 0.50074003        nan 0.50064003 0.50060003
 0.50059003 0.50060003        nan        nan        nan        nan
 0.50064003 0.50060003        nan 0.50060003        nan        nan
 0.50064003        nan 0.50064003 0.50060003 0.50059003 0.50060003
        nan        nan        nan        nan 0.50064003 0.50060003
        nan 0.50060003        nan        nan 0.50064003        nan
 0.50064003 0.50060003 0.50060003 0.50060003        nan        nan
        nan        nan 0.50064003 0.50060003        nan 0.50060003
        nan        nan 0.50064003        nan 0.50064003 0.50060003
 0.50060003 0.50060003        nan        nan        nan        nan
 0.50064003 0.50060003        nan 0.50060003        nan        nan
 0.50059003        nan 0.50064003 0.50060003 0.50060003 0.50060003
        nan        nan        nan        nan 0.50064003 0.50060003
        nan 0.50060003        nan        nan 0.50059003        nan
 0.50064003 0.50060003 0.50060003 0.50060003        nan        nan
        nan        nan 0.50064003 0.50060003        nan 0.50060003
        nan        nan 0.50059003        nan 0.50064003 0.50060003
 0.50060003 0.50060003        nan        nan        nan        nan
 0.50064003 0.50060003        nan 0.50060003        nan        nan
 0.50059003        nan 0.50064003 0.50060003 0.50060003 0.50060003
        nan        nan        nan        nan 0.50064003 0.50060003
        nan 0.50060003        nan        nan 0.50060003        nan
 0.50064003 0.50060003 0.50060003 0.50060003        nan        nan
        nan        nan 0.50064003 0.50060003        nan 0.50060003
        nan        nan 0.50060003        nan 0.50064003 0.50060003
 0.50060003 0.50060003        nan        nan        nan        nan
 0.50064003 0.50060003        nan 0.50060003        nan        nan
 0.50060003        nan 0.50064003 0.50060003 0.50060003 0.50060003
        nan        nan        nan        nan 0.50064003 0.50060003
        nan 0.50060003]
  warnings.warn(
```

In [62]:
```python
best_clf.best_estimator_
```

Out[62]:
```
LogisticRegression(C=0.0001, penalty='l1', random_state=0, solver='liblinear')
```

In [63]:
```python
print (f'Accuracy - : {best_clf.score(x,y):.3f}')
```

```
Accuracy - : 0.502
```

In [ ]: