1. Download the dataset: Dataset

2. Load the dataset.

In [1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

In [8]:
```
data = pd.read_csv("C:/Users/ASUS/Downloads/Churn_Modelling.csv")
data
```

Out[8]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSa
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	10134
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	11254
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	11393
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	9382
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	7908
...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	9627
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	10169
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	4208
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	9288
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	3819

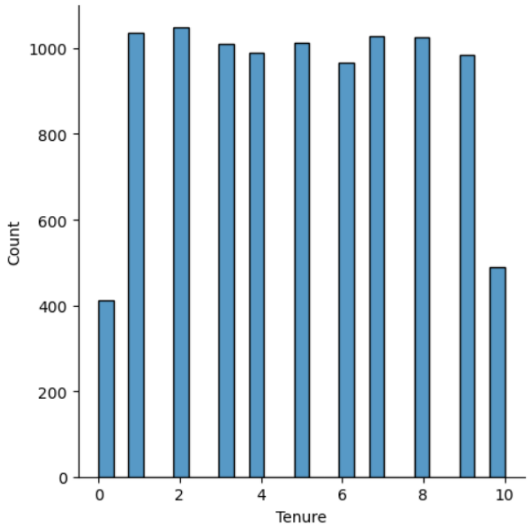
10000 rows × 14 columns

3. Perform Below Visualizations.

- Univariate Analysis

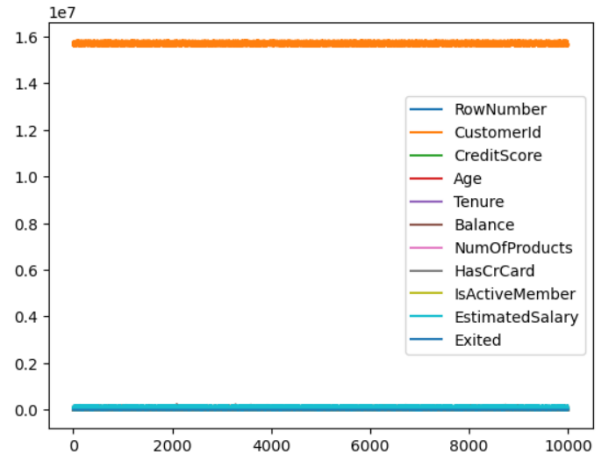
In [9]:
```
sns.displot(data.Tenure)
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x2b69a07e8e0>



- Bi - Variate Analysis

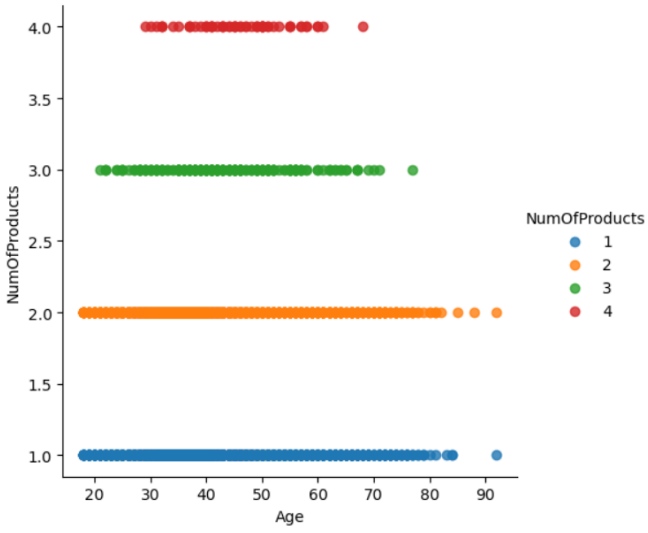
```
In [10]: data.plot.line()
Out[10]: <AxesSubplot:>
```



- Multi - Variate Analysis

```
In [11]: sns.lmplot("Age","NumOfProducts",data,hue="NumOfProducts", fit_reg=False);
```

```
C:\Users\ASUS\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y, data. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an exp
licit keyword will result in an error or misinterpretation.
  warnings.warn(
```



4. Perform descriptive statistics on the dataset.

Age

```
In [12]: data.describe()
```

Out[12]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	1(
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	

5. Handle the Missing values.

```
In [13]: Data = pd.read_csv("Churn_Modelling.csv")
         pd.isnull(Data["Gender"])
```

```
Out[13]: 0       False
         1       False
         2       False
         3       False
         4       False
                 ...
         9995    False
         9996    False
         9997    False
         9998    False
         9999    False
         Name: Gender, Length: 10000, dtype: bool
```

6. Find the outliers and replace the outliers

```
In [14]: data["Tenure"] = np.where(data["Tenure"] >10, np.median,data["Tenure"])
         data["Tenure"]
```

```
Out[14]: 0        2
         1        1
         2        8
         3        1
         4        2
                 ..
         9995     5
         9996    10
         9997     7
         9998     3
         9999     4
         Name: Tenure, Length: 10000, dtype: object
```

‘7. Check for Categorical columns and perform encoding.

```
In [16]: pd.get_dummies(data, columns=["Gender", "Age"], prefix=["Age", "Gender"]).head()
```

Out[16]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	...	Gender_78	Gender_79	G
0	1	15634602	Hargrave	619	France	2	0.00	1	1	1	...	0	0	
1	2	15647311	Hill	608	Spain	1	83807.86	1	0	1	...	0	0	
2	3	15619304	Onio	502	France	8	159660.80	3	1	0	...	0	0	
3	4	15701354	Boni	699	France	1	0.00	2	0	0	...	0	0	
4	5	15737888	Mitchell	850	Spain	2	125510.82	1	1	1	...	0	0	

5 rows × 84 columns

8. Split the data into dependent and independent variables.

Dependent

```
In [17]: x = df.iloc[:, -1].values
         print(x)
```

```
[1 0 1 ... 1 1 0]
```

Independent

```
In [18]: y = data.iloc[:, :-2].values
         print(y)
```

```
[[1 15634602 'Hargrave' ... 1 1 1]
 [2 15647311 'Hill' ... 1 0 1]
 [3 15619304 'Onio' ... 3 1 0]
 ...
 [9998 15584532 'Liu' ... 1 0 1]
 [9999 15682355 'Sabbatini' ... 2 1 0]
 [10000 15628319 'Walker' ... 1 1 0]]
```

9. Scale the independent variables

In [19]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data[["RowNumber"]] = scaler.fit_transform(df[["RowNumber"]])
print(data)
```

```
      RowNumber  CustomerId    Surname  CreditScore Geography  Gender  Age  \
0        0.0000    15634602   Hargrave          619    France  Female   42
1        0.0001    15647311       Hill          608     Spain  Female   41
2        0.0002    15619304       Onio          502    France  Female   42
3        0.0003    15701354       Boni          699    France  Female   39
4        0.0004    15737888   Mitchell          850     Spain  Female   43
...         ...         ...        ...          ...       ...     ...  ...
9995     0.9996    15606229   Obijiaku          771    France    Male   39
9996     0.9997    15569892  Johnstone          516    France    Male   35
9997     0.9998    15584532        Liu          709    France  Female   36
9998     0.9999    15682355  Sabbatini          772   Germany    Male   42
9999     1.0000    15628319     Walker          792    France  Female   28

     Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0         2       0.00              1          1               1
1         1   83807.86              1          0               1
2         8  159660.80              3          1               0
3         1       0.00              2          0               0
4         2  125510.82              1          1               1
...     ...        ...            ...        ...             ...
9995      5       0.00              2          1               0
9996     10   57369.61              1          1               1
9997      7       0.00              1          0               1
9998      3   75075.31              2          1               0
9999      4  130142.79              1          1               0

      EstimatedSalary  Exited
0           101348.88       1
1           112542.58       0
2           113931.57       1
3            93826.63       0
4            79084.10       0
...               ...     ...
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
```

10. Split the data into training and testing

In [21]:
```
from sklearn.model_selection import train_test_split
train_size=0.8
X = df.drop(columns = ['Tenure']).copy()
Y = df['Tenure']
X_train, X_rem, Y_train, Y_rem = train_test_split(X,Y, train_size=0.8)
test_size = 0.5
X_valid, X_test, Y_valid, Y_test = train_test_split(X_rem,Y_rem, test_size=0.5)
print(X_train.shape), print(Y_train.shape)
print(X_valid.shape), print(Y_valid.shape)
print(X_test.shape), print(Y_test.shape)
```

```
(8000, 13)
(8000,)
(1000, 13)
(1000,)
(1000, 13)
(1000,)
```

Out[21]:
```
(None, None)
```