In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [27]:

```python
data = pd.read_csv('customer_churn.csv')
data.head(5)
```

Out[27]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Interr
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	
4	9237-HQITU	Female	0	No	No	2	Yes	No	

5 rows × 21 columns

In [3]:

```python
#A) Data Manipulation:
#a) Extract the 5th column & store it in ‘customer_5’
customer_5=data.iloc[:,4]
customer_5.head()
```

Out[3]:

```
0    No
1    No
2    No
3    No
4    No
Name: Dependents, dtype: object
```

In [4]:

```python
#b. Extract the 15th column & store it in ‘customer_15’
customer_15=data.iloc[:,14]
customer_15.head()
```

Out[4]:

```
0    No
1    No
2    No
3    No
4    No
Name: StreamingMovies, dtype: object
```

In [5]:

```python
#c. Extract all the male senior citizens whose Payment Method is Electronic check & sto
#result in ‘senior_male_electronic’
senior_male_electronic=data[(data['gender']=='Male') & (data['SeniorCitizen'] ==1) & (d
senior_male_electronic
```

Out[5]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
20	8779-QRDMV	Male	1	No	No	1	No	No phone service	
55	1658-BYGOY	Male	1	No	No	18	Yes	Yes	
57	5067-XJQFU	Male	1	Yes	Yes	66	Yes	Yes	
78	0191-ZHSKZ	Male	1	No	No	30	Yes	No	
91	2424-WVHPL	Male	1	No	No	1	Yes	No	
...	...	...	...	...	...	...	...	...	
6837	6229-LSCKB	Male	1	No	No	6	Yes	No	
6894	1400-MMYXY	Male	1	Yes	No	3	Yes	Yes	
6914	7142-HVGBG	Male	1	Yes	No	43	Yes	Yes	
6967	8739-WWKDU	Male	1	No	No	25	Yes	Yes	
7032	6894-LFHLY	Male	1	No	No	1	Yes	Yes	

298 rows × 21 columns

In [6]:

```
#d. Extract all those customers whose tenure is greater than 70 months or their Monthly
#charges is more than 100$ & store the result in ‘customer_total_tenure’
customer_total_tenure=data[(data['tenure']>70) | (data['MonthlyCharges']>100)]
customer_total_tenure
```

Out[6]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	
12	8091-TTVAX	Male	0	Yes	No	58	Yes	Yes	
13	0280-XJGEX	Male	0	No	No	49	Yes	Yes	

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
14	5129-JLPIS	Male	0	No	No	25	Yes	No	
15	3655-SNQYZ	Female	0	Yes	Yes	69	Yes	Yes	
...	...	...	...	...	...	...	...	...	
7023	1035-IPQPU	Female	1	Yes	No	63	Yes	Yes	
7034	0639-TSIQW	Female	0	No	No	67	Yes	Yes	
7037	2569-WGERO	Female	0	No	No	72	Yes	No	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	

1259 rows × 21 columns

In [7]:

```
#e. Extract all the customers whose Contract is of two years, payment method is Mailed
#& the value of Churn is ‘Yes’ & store the result in ‘two_mail_yes’
two_mail_yes=data[(data['Contract']=='Two year') & (data['PaymentMethod']=='Mailed chec
two_mail_yes
```

Out[7]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
268	6323-AYBRX	Male	0	No	No	59	Yes	No	
5947	7951-QKZPL	Female	0	Yes	Yes	33	Yes	Yes	
6680	9412-ARGBX	Female	0	No	Yes	48	Yes	No	

3 rows × 21 columns

In [8]:

```
#f.Extract 333 random records from the customer_churndataframe& store the result in
#‘customer_333’
customer_333=data.sample(n=333)
customer_333
```

Out[8]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
6791	5204-QZXPU	Male	0	No	No	19	No	No phone service	

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	In
1894	9281-PKKZE	Female	0	Yes	No	46	No	No phone service	
6905	4459-BBGHE	Male	0	No	Yes	30	No	No phone service	
3695	6088-BXMRG	Female	0	Yes	Yes	32	Yes	Yes	
3631	2722-JMONI	Female	1	Yes	No	1	Yes	No	
...	...	...	...	...	...	...	...	...	
1557	4672-FOTSD	Male	0	No	No	12	Yes	Yes	
3352	9124-LHCJQ	Female	0	No	No	1	Yes	Yes	
3667	7826-VVKWT	Female	1	Yes	Yes	24	Yes	No	
2675	4878-BUNFV	Male	0	Yes	Yes	42	Yes	No	
364	3583-KRKMD	Male	0	No	No	18	Yes	No	

333 rows × 21 columns

In [9]:

```
#g. Get the count of different levels from the ‘Churn’ column
data['Churn'].value_counts()
```

Out[9]:

```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

In [10]:

```
#B) Data Visualization:
#a. Build a bar-plot for the ’InternetService’ column:
#i. Set x-axis label to ‘Categories of Internet Service’
#ii. Set y-axis label to ‘Count of Categories’
#iii. Set the title of plot to be ‘Distribution of Internet Service’
#iv. Set the color of the bars to be ‘orange’
```

In [11]:

```
data['InternetService'].value_counts()
```

Out[11]:

```
Fiber optic    3096
DSL            2421
No             1526
Name: InternetService, dtype: int64
```

In [12]:

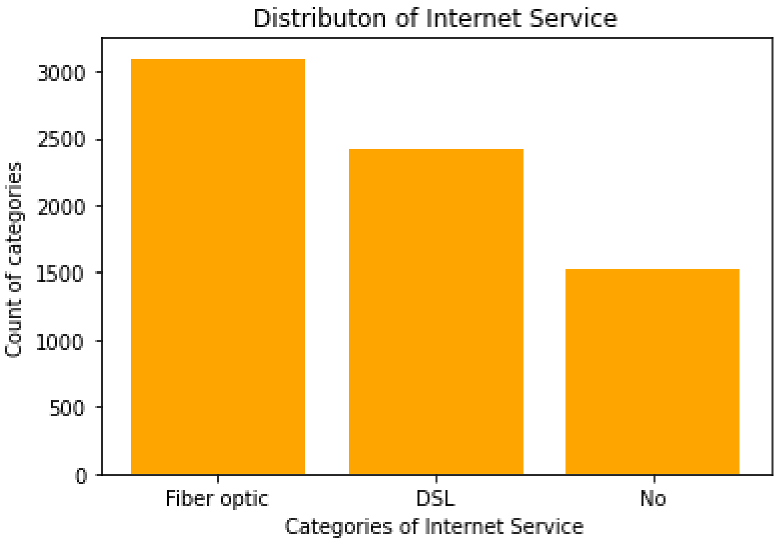
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
x=data['InternetService'].value_counts().keys().tolist()
y=data['InternetService'].value_counts().tolist()
```

In [13]:

```
plt.bar(x,y, color='orange')
plt.xlabel('Categories of Internet Service')
plt.ylabel('Count of categories')
plt.title('Distributon of Internet Service')
```

Out[13]:

```
Text(0.5, 1.0, 'Distributon of Internet Service')
```

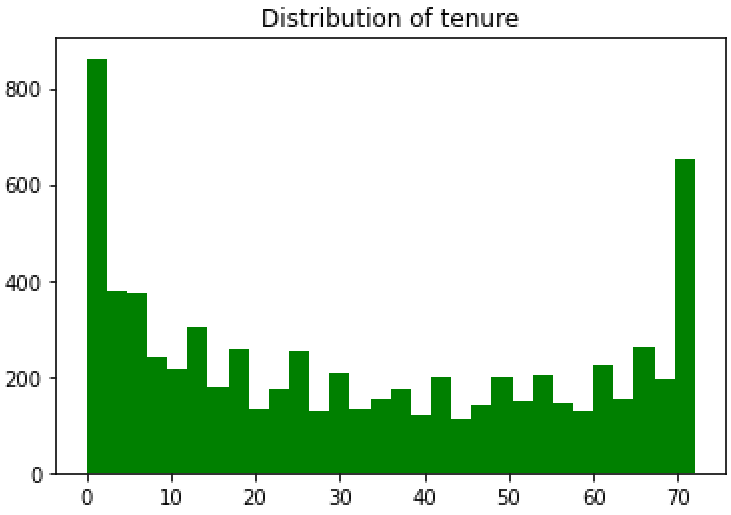


In [14]:

```
#b. Build a histogram for the ‘tenure’ column:
#i. Set the number of bins to be 30
#ii. Set the color of the bins to be ‘green’
#iii. Assign the title ‘Distribution of tenure’
plt.hist(data['tenure'],color='green', bins=30)
plt.title('Distribution of tenure')
```

Out[14]:

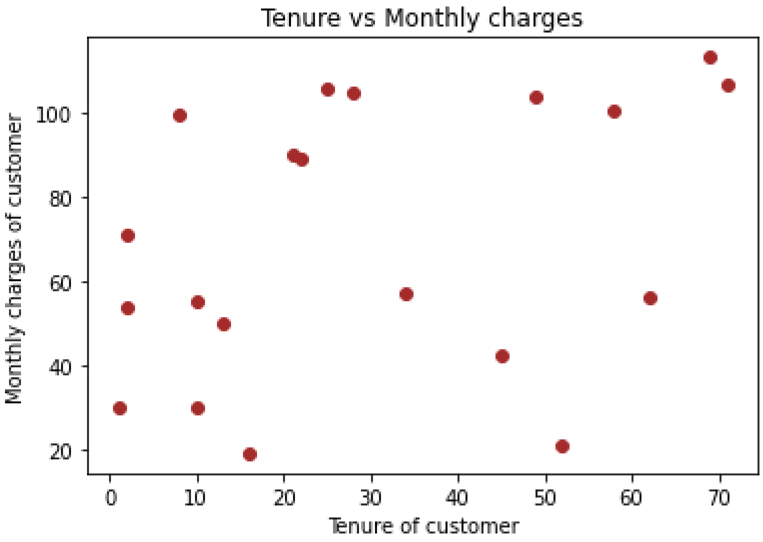
```
Text(0.5, 1.0, 'Distribution of tenure')
```



In [15]:

```
#c. Build a scatter-plot between ‘MonthlyCharges’ & ‘tenure’. Map ‘MonthlyCharges’ to t
#& ‘tenure’ to the ‘x-axis’:
#i. Assign the points a color of ‘brown’
```

```
#ii. Set the x-axis label to ‘Tenure of customer’
#iii. Set the y-axis label to ‘Monthly Charges of customer’
#iv. Set the title to ‘Tenure vs Monthly Charges’
plt.scatter(x=data['tenure'].head(20),y=data['MonthlyCharges'].head(20), color='Brown')
plt.xlabel('Tenure of customer')
plt.ylabel('Monthly charges of customer')
plt.title('Tenure vs Monthly charges')
plt.show()
```

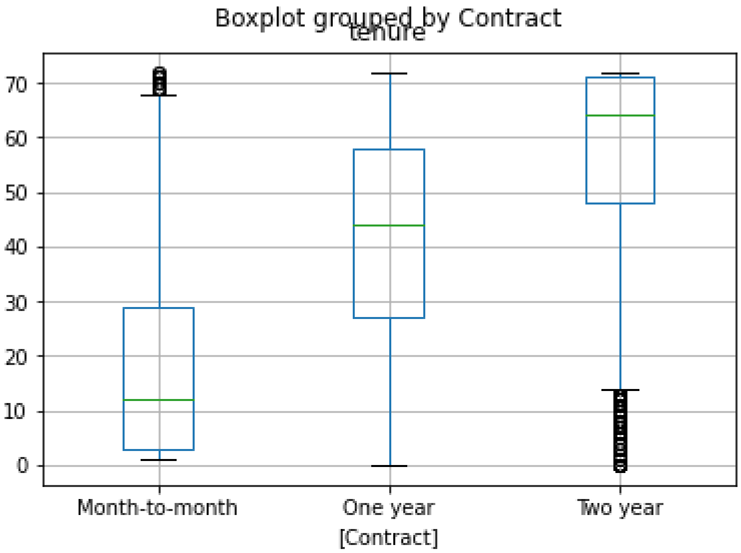


In [16]:

```
#d. Build a box-plot between ‘tenure’ & ‘Contract’. Map ‘tenure’ on the y-axis & ‘Contr
#the x-axis.
data.boxplot(column='tenure',by=['Contract'])
```

Out[16]:

```
<AxesSubplot:title={'center':'tenure'}, xlabel='[Contract]'>
```



In [28]:

```
data.Churn=data.Churn.map(dict(Yes=1, No=0))
```

In [29]:

```
#C) Linear Regression:
#a. Build a simple linear model where dependent variable is ‘Churn’ and independent
#variable is ‘tenure’
#data =pd.read_csv('customer_churn.csv')
```

```
x = data.loc[:,['tenure']].values
y = data.loc[:,['Churn']].values
```

In [30]:
```
x
```

Out[30]:
```
array([[ 1],
       [34],
       [ 2],
       ...,
       [11],
       [ 4],
       [66]], dtype=int64)
```

In [31]:
```
y
```

Out[31]:
```
array([[0],
       [0],
       [1],
       ...,
       [0],
       [1],
       [0]], dtype=int64)
```

In [ ]:

In [32]:
```
#i. Divide the dataset into train and test sets in 70:30 ratio.
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test=train_test_split(x,y,train_size=0.7,random_state=0)
```

In [33]:
```
print(x_train)
print(y_train)
print(x_test)
print(x_test)
```

```
[[ 9]
 [14]
 [64]
 ...
 [58]
 [ 1]
 [ 4]]
[[1]
 [0]
 [1]
 ...
 [0]
 [1]
 [0]]
[[19]
 [60]
 [13]
 ...
 [69]
 [52]
 [35]]
[[19]
 [60]
 [13]
 ...
 [69]
 [52]
 [35]]
```

In [34]:

```
#ii. Build the model on train set and predict the values on test set
from sklearn.linear_model import LinearRegression
simpleLinearRegression = LinearRegression()
simpleLinearRegression.fit(x_train, y_train)
```

Out[34]:

```
LinearRegression()
```

In [35]:

```
#iii. After predicting the values, find the root mean square error
y_pred = simpleLinearRegression.predict(x_test)
y_pred
```

Out[35]:

```
array([[0.35471089],
       [0.08617374],
       [0.39400901],
       ...,
       [0.02722656],
       [0.13857123],
       [0.24991591]])
```

In [36]:

```
#D) Logistic Regression:
#a. Build a simple logistic regression modelwhere dependent variable is ‘Churn’ & indep
#variable is ‘MonthlyCharges’
y= data.loc[:,['Churn']].values
x= data.loc[:,['MonthlyCharges']].values
```

In [37]:

```
#i. Divide the dataset in 65:35 ratio
x_train,x_test, y_train,y_test=train_test_split(x,y,train_size=0.65, random_state=0)
```

In [38]:

```
#ii. Build the model on train set and predict the values on test set
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(x_train,y_train)
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversio
nWarning: A column-vector y was passed when a 1d array was expected. Please change the s
hape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

Out[38]:

```
LogisticRegression()
```

In [39]:

```
y_pred= logmodel.predict(x_test)
y_pred
```

Out[39]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [40]:

```
y_test
```

Out[40]:

```
array([[0],
       [0],
       [0],
       ...,
       [1],
       [0],
       [0]], dtype=int64)
```

In [41]:

```
#iii. Build the confusion matrix and get the accuracy score
from sklearn.metrics import confusion_matrix,accuracy_score
```

```
print(confusion_matrix(y_pred,y_test))
print (accuracy_score(y_pred,y_test))
```

```
[[1815  651]
 [   0    0]]
0.7360097323600974
```

In [42]:

```
#E) Decision Tree:
#a. Build a decision tree model where dependent variable is ‘Churn’ & independent varia
#‘tenure’
x=data.loc[:,['tenure']].values
y=data.loc[:,['Churn']].values
```

In [43]:

```
#i. Divide the dataset in 80:20 ratio
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= 0.20, random_state
```

In [44]:

```
#ii. Build the model on train set and predict the values on test set

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(x_train, y_train)
```

Out[44]:

```
DecisionTreeClassifier()
```

In [45]:

```
y_pred = classifier.predict(x_test)
y_pred
```

Out[45]:

```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [46]:

```
#iii. Build the confusion matrix and calculate the accuracy

from sklearn.metrics import classification_report, confusion_matrix,accuracy_score
print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

```
[[965  76]
 [281  87]]
0.7466288147622427
```

In [47]:

```
#F) Random Forest:
#a. Build a Random Forest model where dependent variable is ‘Churn’ & independent varia
#are ‘tenure’ and ‘MonthlyCharges’

x=data.loc[:,['tenure', 'MonthlyCharges']].values
y=data.loc[:,'Churn'].values
```

In [48]:

```
#i. Divide the dataset in 70:30 ratio
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= 0.30, random_state
```

In [49]:

```
#ii. Build the model on train set and predict the values on test set
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=200)
clf.fit(x_train,y_train)
```

Out[49]:

```
RandomForestClassifier(n_estimators=200)
```

In [50]:

```
y_pred=clf.predict(x_test)
```

In [51]:

```
#iii. Build the confusion matrix and calculate the accuracy
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("The confusion matrix:",metrics.confusion_matrix(y_pred,y_test))
```

```
Accuracy: 0.7482252721249408
The confusion matrix: [[1351  323]
 [ 209  230]]
```

In [ ]:

In [ ]: