EDA-01 Know Your Data

1. Necessary Imports

In [4]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

a) What is Panadas in Python?

Pandas is an open source Python library which provides necessray support for implementing Sereis and dataframe data structures. A Pandas Sereis is a one dimensional array with homogenious data . The size of sereis is immutable but values are mutable. A dataframe is a data structure to hold two dimensional array with hetrogenious data columns. Pandas Dataframe is the ideal data structure for EDA and Pre-Processing Tasks. Pandas Dataframe Features:

1. Reading different file types like Excel and CSV
2. Default and user defined indexing
3. Easy data access based on indices
4. Easy handling of Missing Values
5. Easy Data Splitting & Merging , Concatenation and integration
6. Easy data sorting and aggregation
7. Easy and efficient handling of time sereis data

b) Numpy

Numpy stands for numerical python and provides support for efficient mathematical operations on array type data

c) Data Dictionary

A data dictionary is a data structure in python which stores data in the form of key-value pairs.

d) matplotlib and seaborn

Matplolib and seaborn are python graphic libraries . Matplotlib is used for basic plotting while seaborn which is based on matplotlib is used for advance statistical visualizations.

2.Creating a Data Dictionary

In [5]:
```
data={'Name':['Ali','Ahmed','Sara','Tom','Mike','Maria'],
      'Gender':['Male','Male','Female','Male','Male','Female'],
      'Age':[20,30, 15, 24, 23, 20],
      'Height':[5.2,5.4,5.1,5.5,5.6,5.0],
      'Play':[True,True,False,True,False,True]}
```

In [6]:
```
data
```

Out[6]:
```
{'Name': ['Ali', 'Ahmed', 'Sara', 'Tom', 'Mike', 'Maria'],
 'Gender': ['Male', 'Male', 'Female', 'Male', 'Male', 'Female'],
 'Age': [20, 30, 15, 24, 23, 20],
 'Height': [5.2, 5.4, 5.1, 5.5, 5.6, 5.0],
 'Play': [True, True, False, True, False, True]}
```

In [7]:
```
type(data)
```

Out[7]:
```
dict
```

3. Converting a Data Dictionary into a Dataframe

In [8]:
```
data=pd.DataFrame(data)
```

In [9]:
```
data
```

Out[9]:

	Name	Gender	Age	Height	Play
0	Ali	Male	20	5.2	True
1	Ahmed	Male	30	5.4	True
2	Sara	Female	15	5.1	False
3	Tom	Male	24	5.5	True
4	Mike	Male	23	5.6	False
5	Maria	Female	20	5.0	True

4.Some Basic Commands to Describe a Dataframe

i) shape()

In [10]:
```
data.shape
```

Out[10]:
```
(6, 5)
```

There are 6 rows and five columns- There is an andex as well

ii) columns

In [11]:
```
data.columns
```

Out[11]:
```
Index(['Name', 'Gender', 'Age', 'Height', 'Play'], dtype='object')
```

data.columns shows the column names

iii) info()

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Name    6 non-null      object 
 1   Gender  6 non-null      object 
 2   Age     6 non-null      int64  
 3   Height  6 non-null      float64
 4   Play    6 non-null      bool   
dtypes: bool(1), float64(1), int64(1), object(2)
memory usage: 326.0+ bytes
```

info()command displays a detailed information about the dataset including :

1. RangeIndex (Range of Index)
2. Data Columns (Total number of columns)
3. List of all columns with column index , name , Count of Non Null Values and data ty
pe
4. Information about data types
5. Memory usage

iv). head() and tail()

In [13]:
```
data.head()
```

Out[13]:

	Name	Gender	Age	Height	Play
0	Ali	Male	20	5.2	True
1	Ahmed	Male	30	5.4	True
2	Sara	Female	15	5.1	False
3	Tom	Male	24	5.5	True
4	Mike	Male	23	5.6	False

head()shows 5 records from the top whereas tail()shows 5 records from the tail. we can give a paramater to indicate how many rows are required.

In [14]:
```
data.head(3)
```

Out[14]:

	Name	Gender	Age	Height	Play
0	Ali	Male	20	5.2	True
1	Ahmed	Male	30	5.4	True
2	Sara	Female	15	5.1	False

In [15]:
```
data.tail(3)
```

Out[15]:

	Name	Gender	Age	Height	Play
3	Tom	Male	24	5.5	True
4	Mike	Male	23	5.6	False
5	Maria	Female	20	5.0	True

v) describe()

In [16]:
```
data.describe()
```

Out[16]:

	Age	Height
count	6.00000	6.000000
mean	22.00000	5.300000
std	5.01996	0.236643
min	15.00000	5.000000
25%	20.00000	5.125000
50%	21.50000	5.300000
75%	23.75000	5.475000
max	30.00000	5.600000

describe()command shows some statistical measures including count, mean, standard deviation, minimum, Q1, Q2(Median), Q3 and maximum for all the numeric attributes. By default it does not show other types of attributes. we can include them by defining the include paramater.

In [17]:
```
data.describe(include=['object'])
```

Out[17]:

	Name	Gender
count	6	6
unique	6	2
top	Mike	Male
freq	1	4

In [18]:
```
data.describe(include=['object', 'integer','bool','float'])
```

Out[18]:

	Name	Gender	Age	Height	Play
count	6	6	6.00000	6.000000	6
unique	6	2	NaN	NaN	2
top	Mike	Male	NaN	NaN	True
freq	1	4	NaN	NaN	4
mean	NaN	NaN	22.00000	5.300000	NaN
std	NaN	NaN	5.01996	0.236643	NaN
min	NaN	NaN	15.00000	5.000000	NaN
25%	NaN	NaN	20.00000	5.125000	NaN
50%	NaN	NaN	21.50000	5.300000	NaN
75%	NaN	NaN	23.75000	5.475000	NaN
max	NaN	NaN	30.00000	5.600000	NaN

vi) dtypes

In [19]:
```
data.dtypes
```

Out[19]:
```
Name       object
Gender     object
Age         int64
Height    float64
Play         bool
dtype: object
```

dtypes shows the data types of all the columns in the dataframe

vii) Some commands to get basic statistical measures for individual columns

In [20]:
```
data['Age'].mean()
```

Out[20]: 22.0

In [21]:
```
data['Age'].median()
```

Out[21]: 21.5

In [22]:
```
data['Age'].min()
```

Out[22]: 15

In [23]:
```
data['Age'].max()
```

Out[23]: 30

In [24]:
```
data['Age'].var()
```

Out[24]: 25.2

In [25]:
```
Q1=np.quantile(data['Age'],0.25)
Q1
```

Out[25]: 20.0

In [26]:
```
Q2=np.quantile(data['Age'],0.50)
Q2
```

Out[26]: 21.5

In [27]:
```
Q3=np.quantile(data['Age'],0.75)
Q3
```

Out[27]: 23.75

5). Reading CSV/ Excel files into a Dataframe

In [28]:
```
data=pd.read_csv("D:/FTI/Cohort 2 EDA/Lecture 1/sampleCSV.csv")
```

In [29]:
```
data.head()
```

Out[29]:

	S.NO	Name	Age	Gender
0	1	Ali	20	Male
1	2	Ahmed	21	Male
2	3	Usman	22	Male
3	4	Akram	23	Male
4	5	Tom	19	Male

In [30]:
```
data=pd.read_csv("D:/FTI/Cohort 2 EDA/Lecture 1/sampleCSVNoHeader.csv", header=None)
```

In [31]:
```
data
```

Out[31]:

	0	1	2	3
0	1	Ali	20	Male
1	2	Ahmed	21	Male
2	3	Usman	22	Male
3	4	Akram	23	Male
4	5	Tom	19	Male
5	6	Sara	17	Female
6	7	David	23	Male
7	8	Dravid	22	Male
8	9	Bob	18	Male
9	10	Clara	21	Female

In [32]:
```
data=pd.read_excel("D:/FTI/Cohort 2 EDA/Lecture 1/sampleExcel.xlsx")
```

In [33]:
```
data.head()
```

Out[33]:

	S.NO	Name	Age	Gender
0	1	Ali	20	Male
1	2	Ahmed	21	Male
2	3	Usman	22	Male
3	4	Akram	23	Male
4	5	Tom	19	Male

In [34]:
```
data=pd.read_excel("D:/FTI/Cohort 2 EDA/Lecture 1/sampleExcelNoHeader.xlsx", header
=None)
```

In [35]:
```
data.head()
```

Out[35]:

	0	1	2	3
0	1	Ali	20	Male
1	2	Ahmed	21	Male
2	3	Usman	22	Male
3	4	Akram	23	Male
4	5	Tom	19	Male

In [36]:
```
data.columns=['ID','Name','Age','Gender']
```

In [37]:
```
data.head()
```

Out[37]:

	ID	Name	Age	Gender
0	1	Ali	20	Male
1	2	Ahmed	21	Male
2	3	Usman	22	Male
3	4	Akram	23	Male
4	5	Tom	19	Male

6. Saving a dataframe as a csv

In [38]:
```
data.to_csv("D:/FTI/Cohort 2 EDA/Lecture 1/data.csv", index=False)
```

7. Working with Telecom Churn Dataset

i) Load the dataset

In [39]:
```
data=pd.read_csv("D:/FTI/Cohort 2 EDA/Lecture 1/TelecomChurn.csv")
```

In [40]:
```
data.head()
```

Out[40]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	r
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	

5 rows × 21 columns

ii) Basis Data Description

In [41]:
```
data.shape
```

Out[41]: (3333, 21)

In [42]:
```
pd.set_option('Display.Max_Columns',50)
```

In [43]:
```
data.head()
```

Out[43]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	t cha
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	197.4	99	16
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	195.5	103	16
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	121.2	110	10
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	61.9	88	5
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	148.3	122	12

In [44]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   state                   3333 non-null   object 
 1   account length          3333 non-null   int64  
 2   area code               3333 non-null   int64  
 3   phone number            3333 non-null   object 
 4   international plan      3333 non-null   object 
 5   voice mail plan         3333 non-null   object 
 6   number vmail messages   3333 non-null   int64  
 7   total day minutes       3333 non-null   float64
 8   total day calls         3333 non-null   int64  
 9   total day charge        3333 non-null   float64
 10  total eve minutes       3333 non-null   float64
 11  total eve calls         3333 non-null   int64  
 12  total eve charge        3333 non-null   float64
 13  total night minutes     3333 non-null   float64
 14  total night calls       3333 non-null   int64  
 15  total night charge      3333 non-null   float64
 16  total intl minutes      3333 non-null   float64
 17  total intl calls        3333 non-null   int64  
 18  total intl charge       3333 non-null   float64
 19  customer service calls  3333 non-null   int64  
 20  churn                   3333 non-null   bool   
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 524.2+ KB
```

In [45]:
```
data.dtypes
```

Out[45]:
```
state                     object
account length             int64
area code                  int64
phone number              object
international plan        object
voice mail plan           object
number vmail messages      int64
total day minutes        float64
total day calls            int64
total day charge         float64
total eve minutes        float64
total eve calls            int64
total eve charge         float64
total night minutes      float64
total night calls          int64
total night charge       float64
total intl minutes       float64
total intl calls           int64
total intl charge        float64
customer service calls     int64
churn                       bool
dtype: object
```

In [46]:
```
data.describe()
```

Out[46]:

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total ev call
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.00000
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.980348	100.11431
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.713844	19.92262
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.600000	87.00000
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.400000	100.00000
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.300000	114.00000
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.700000	170.00000

In [47]:
```
data.describe(include=['object'])
```

Out[47]:

	state	phone number	international plan	voice mail plan
count	3333	3333	3333	3333
unique	51	3333	2	2
top	WV	381-7597	no	no
freq	106	1	3010	2411

In [48]:
```
data.describe(include=['bool'])
```

Out[48]:

	churn
count	3333
unique	2
top	False
freq	2850

In [49]:
```
data.head(10)
```

Out[49]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	cha
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	197.4	99	16
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	195.5	103	16
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	121.2	110	10
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	61.9	88	5
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	148.3	122	12
5	AL	118	510	391-8027	yes	no	0	223.4	98	37.98	220.6	101	18
6	MA	121	510	355-9993	no	yes	24	218.2	88	37.09	348.5	108	29
7	MO	147	415	329-9001	yes	no	0	157.0	79	26.69	103.1	94	8
8	LA	117	408	335-4719	no	no	0	184.5	97	31.37	351.6	80	29
9	WV	141	415	330-8173	yes	yes	37	258.6	84	43.96	222.0	111	18

In [50]:
```
data.columns
```

Out[50]:
```
Index(['state', 'account length', 'area code', 'phone number',
       'international plan', 'voice mail plan', 'number vmail messages',
       'total day minutes', 'total day calls', 'total day charge',
       'total eve minutes', 'total eve calls', 'total eve charge',
       'total night minutes', 'total night calls', 'total night charge',
       'total intl minutes', 'total intl calls', 'total intl charge',
       'customer service calls', 'churn'],
      dtype='object')
```

8) Univariare Analysis

i) Ctaegorical and Boolean Attributes

In [51]:
```
data.describe(include=['object', 'bool'])
```

Out[51]:

	state	phone number	international plan	voice mail plan	churn
count	3333	3333	3333	3333	3333
unique	51	3333	2	2	2
top	WV	381-7597	no	no	False
freq	106	1	3010	2411	2850

a) describe()

In [52]:
```
data['state'].describe()
```

Out[52]:
```
count      3333
unique       51
top          WV
freq        106
Name: state, dtype: object
```

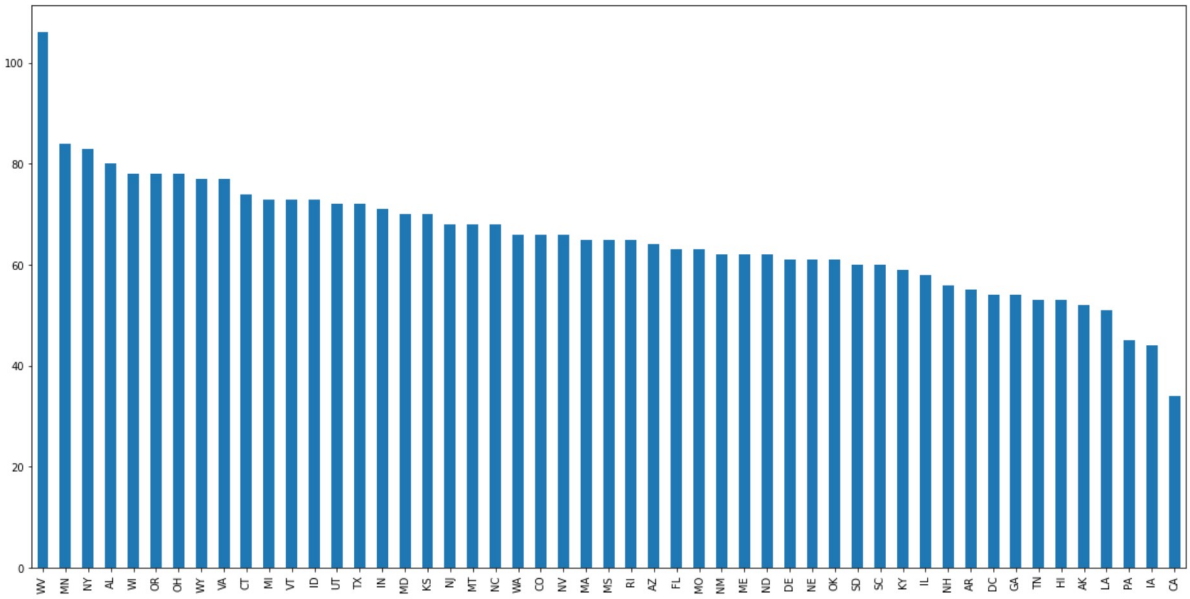
b) value_counts

In [53]:
```
data['state'].value_counts();
```

c) Bar Chart

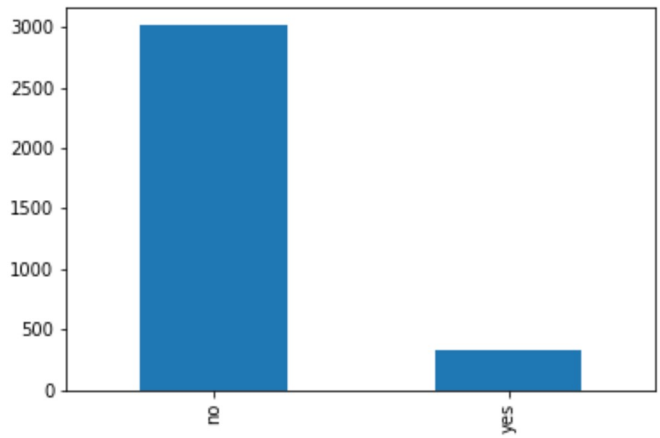
In [171]:
```
data['state'].value_counts().plot(kind='bar', figsize=(20,10))
```

Out[171]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1be37a13f28>
```



In [170]:
```
data['international plan'].value_counts().plot(kind='bar')
```

Out[170]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1be379c1748>
```



ii) Numeric Attributes

In [172]:
```
data.describe()
```

Out[172]:

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total ev call
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.00000
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.980348	100.11431
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.713844	19.92262
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.600000	87.00000
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.400000	100.00000
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.300000	114.00000
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.700000	170.00000

a) describe()

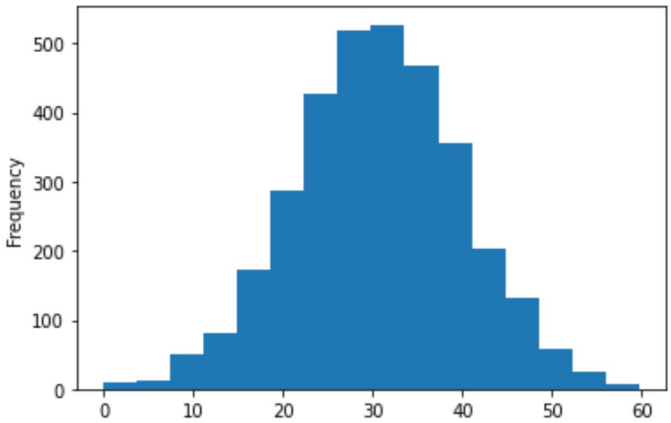
In [173]:
```
data['total day charge'].describe()
```

Out[173]:
```
count    3333.000000
mean       30.562307
std         9.259435
min         0.000000
25%        24.430000
50%        30.500000
75%        36.790000
max        59.640000
Name: total day charge, dtype: float64
```

b) histogram

In [188]:
```
data['total day charge'].plot.hist(bins=16)
```

Out[188]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1be373b6828>
```

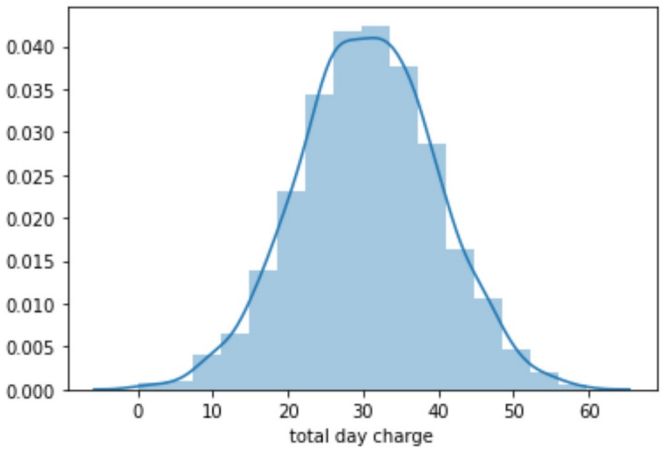


In [54]:
```
import seaborn as sns
```

In [55]:
```
sns.distplot(data['total day charge'],bins=16, kde=True)
```

```
C:\Users\malik\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarn
ing: Using a non-tuple sequence for multidimensional indexing is deprecated; use
`arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted
as an array index, `arr[np.array(seq)]`, which will result either in an error or
a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

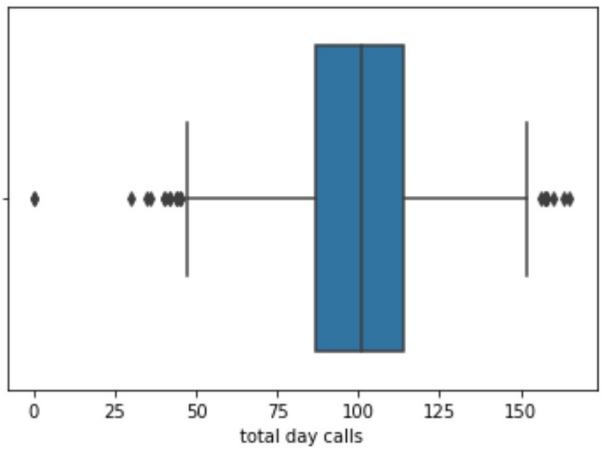
Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22057c5a668>
```



c) boxplot

In [193]:
```
sns.boxplot(x='total day calls', data=data)
```

Out[193]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1be3ab3c320>
```



9. Bivariate Analysis

In [194]:
```
data['churn'].value_counts()
```

Out[194]:
```
False    2850
True      483
Name: churn, dtype: int64
```

In [195]:
```
data.head()
```

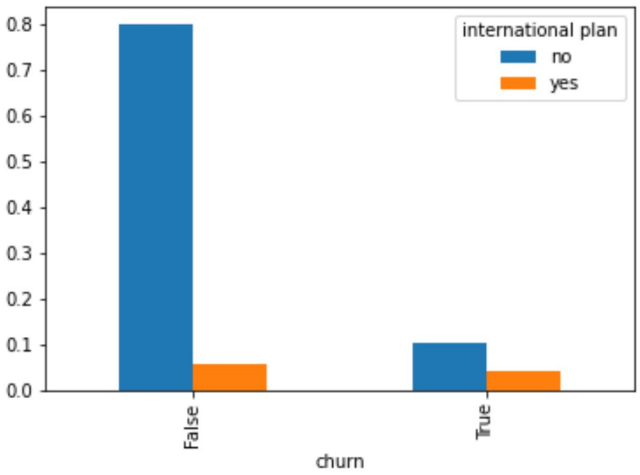
Out[195]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	t cha
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	197.4	99	16
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	195.5	103	16
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	121.2	110	1(
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	61.9	88	5
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	148.3	122	12

a) cross tab [Categorical vs Categorical]

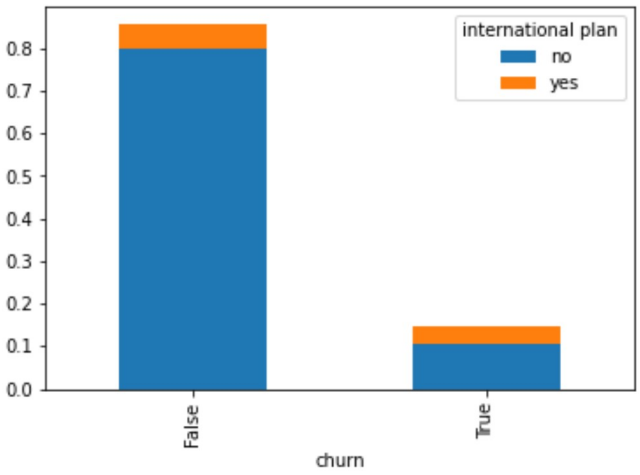
In [199]:
```
pd.crosstab(data['churn'], data['international plan'],normalize=True).plot(kind='bar')
```

Out[199]: <matplotlib.axes._subplots.AxesSubplot at 0x1be3ab9c908>



In [200]:
```
pd.crosstab(data['churn'], data['international plan'],normalize=True).plot(kind='bar', stacked=True)
```

Out[200]: <matplotlib.axes._subplots.AxesSubplot at 0x1be3acb5cc0>



b) Categorical vs Numeric (groupby)

Whether churners make more customer calls then non-churners

In [212]:
```
data.groupby('churn')['customer service calls'].mean()
```

Out[212]:
```
churn
False    1.449825
True     2.229814
Name: customer service calls, dtype: float64
```

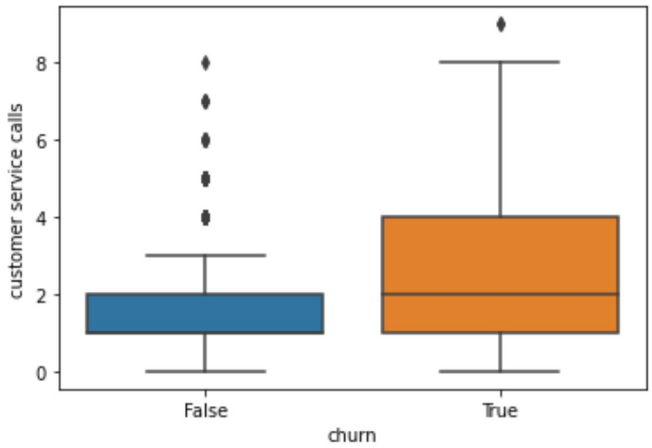
which state has the higest number of churners

In [257]:
```
data.groupby('state')['churn'].value_counts().head(20)
```

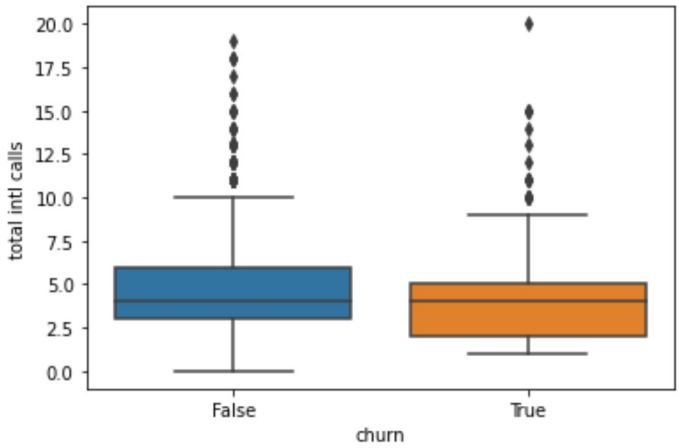
Out[257]:
```
state  churn
AK     False    49
       True      3
AL     False    72
       True      8
AR     False    44
       True     11
AZ     False    60
       True      4
CA     False    25
       True      9
CO     False    57
       True      9
CT     False    62
       True     12
DC     False    49
       True      5
DE     False    52
       True      9
FL     False    55
       True      8
Name: churn, dtype: int64
```

c) boxplots

In [273]:
```
sns.boxplot( x=data["churn"], y=data["customer service calls"] );
plt.show()
```



In [275]:
```
sns.boxplot( x=data["churn"], y=data["total intl calls"] );
plt.show()
```



In [ ]:
```

```