```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("/content/bike_buyers.csv")
df
```

	ID	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	12496	Married	Female	40000.0	1.0	Bachelors	Skilled Manual	Yes	0.0	0-1 Miles	Europe	42.0	No
1	24107	Married	Male	30000.0	3.0	Partial College	Clerical	Yes	1.0	0-1 Miles	Europe	43.0	No
2	14177	Married	Male	80000.0	5.0	Partial College	Professional	No	2.0	2-5 Miles	Europe	60.0	No
3	24381	Single	NaN	70000.0	0.0	Bachelors	Professional	Yes	1.0	5-10 Miles	Pacific	41.0	Yes
4	25597	Single	Male	30000.0	0.0	Bachelors	Clerical	No	0.0	0-1 Miles	Europe	36.0	Yes
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	23731	Married	Male	60000.0	2.0	High School	Professional	Yes	2.0	2-5 Miles	North America	54.0	Yes
996	28672	Single	Male	70000.0	4.0	Graduate Degree	Professional	Yes	0.0	2-5 Miles	North America	35.0	Yes
997	11809	Married	NaN	60000.0	2.0	Bachelors	Skilled Manual	Yes	0.0	0-1 Miles	North America	38.0	Yes
998	19664	Single	Male	100000.0	3.0	Bachelors	Management	No	3.0	1-2 Miles	North America	38.0	No
999	12121	Single	Male	60000.0	3.0	High School	Professional	Yes	2.0	10+ Miles	North America	53.0	Yes

```
df.head()
```

	ID	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	12496	Married	Female	40000.0	1.0	Bachelors	Skilled Manual	Yes	0.0	0-1 Miles	Europe	42.0	No
1	24107	Married	Male	30000.0	3.0	Partial College	Clerical	Yes	1.0	0-1 Miles	Europe	43.0	No
2	14177	Married	Male	80000.0	5.0	Partial College	Professional	No	2.0	2-5 Miles	Europe	60.0	No
3	24381	Single	NaN	70000.0	0.0	Bachelors	Professional	Yes	1.0	5-10 Miles	Pacific	41.0	Yes
4	25597	Single	Male	30000.0	0.0	Bachelors	Clerical	No	0.0	0-1 Miles	Europe	36.0	Yes

```
df.tail()
```

	ID	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
995	23731	Married	Male	60000.0	2.0	High School	Professional	Yes	2.0	2-5 Miles	North America	54.0	Yes
996	28672	Single	Male	70000.0	4.0	Graduate Degree	Professional	Yes	0.0	2-5 Miles	North America	35.0	Yes
997	11809	Married	NaN	60000.0	2.0	Bachelors	Skilled Manual	Yes	0.0	0-1 Miles	North America	38.0	Yes
998	19664	Single	Male	100000.0	3.0	Bachelors	Management	No	3.0	1-2 Miles	North America	38.0	No
999	12121	Single	Male	60000.0	3.0	High School	Professional	Yes	2.0	10+ Miles	North America	53.0	Yes

```
df.shape
```

```
(1000, 13)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                1000 non-null   int64
 1   Marital Status    993 non-null    object
 2   Gender            989 non-null    object
 3   Income            994 non-null    float64
 4   Children          992 non-null    float64
 5   Education         1000 non-null   object
 6   Occupation        1000 non-null   object
 7   Home Owner        996 non-null    object
 8   Cars              991 non-null    float64
 9   Commute Distance  1000 non-null   object
 10  Region            1000 non-null   object
 11  Age               992 non-null    float64
 12  Purchased Bike    1000 non-null   object
dtypes: float64(4), int64(1), object(8)
memory usage: 101.7+ KB
```

```
df.describe()
```

	ID	Income	Children	Cars	Age
count	1000.000000	994.000000	992.000000	991.000000	992.000000
mean	19965.992000	56267.605634	1.910282	1.455096	44.181452
std	5347.333948	31067.817462	1.626910	1.121755	11.362007
min	11000.000000	10000.000000	0.000000	0.000000	25.000000
25%	15290.750000	30000.000000	0.000000	1.000000	35.000000
50%	19744.000000	60000.000000	2.000000	1.000000	43.000000
75%	24470.750000	70000.000000	3.000000	2.000000	52.000000
max	29447.000000	170000.000000	5.000000	4.000000	89.000000

```
df.isnull()
```

	ID	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	True	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False	False	False	False	False	False	False
996	False	False	False	False	False	False	False	False	False	False	False	False	False
997	False	False	True	False	False	False	False	False	False	False	False	False	False
998	False	False	False	False	False	False	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False	False	False	False	False	False	False

```
df.isnull().sum()
```

	0
ID	0
Marital Status	7
Gender	11
Income	6
Children	8
Education	0
Occupation	0
Home Owner	4
Cars	9
Commute Distance	0
Region	0
Age	8
Purchased Bike	0

```
df.isnull().mean()
```

	0
ID	0.000
Marital Status	0.007
Gender	0.011
Income	0.006
Children	0.008
Education	0.000
Occupation	0.000
Home Owner	0.004
Cars	0.009
Commute Distance	0.000
Region	0.000
Age	0.008
Purchased Bike	0.000

```
df["Marital Status"].to_string()
```

```
'0      Married\n1      Married\n2      Married\n3       Single\n4       Single\n5      Married\n6       Single\n7      Married\n8
NaN\n9      Married\n10      Married\n11       Single\n12      Married\n13      Married\n14       Single\n15       Single\n16       Single\n17
Married\n18       Single\n19       Single\n20      Married\n21       Single\n22       Single\n23      Married\n24       Single\n25       Single
\n26       Single\n27          NaN\n28      Married\n29       Single\n30      Married\n31      Married\n32       Single\n33       Single\n34
Single\n35       Single\n36      Married\n37       Single\n38       Single\n39       Single\n40       Single\n41       Single\n42      Married
\n43      Married\n44      Married\n45      Married\n46      Married\n47       Single\n48      Married\n49          NaN\n50       Single\n51
Single\n52      Married\n53       Single\n54       Single\n55      Married\n56      Married\n57      Married\n58      Married\n59      Married
```

CATEGORICAL DATA FILLING

Filling Data by Mode method.

```
# Percentage of null values in Marital Status.
df["Marital Status"].isnull().mean()
```

```
0.007
```

```
# Filling data
df["Marital Status"].fillna(df["Marital Status"].mode()[0],inplace=True)
```

```
<ipython-input-78-f2ee6d380762>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df["Marital Status"].fillna(df["Marital Status"].mode()[0],inplace=True)
```

```
df["Marital Status"].isnull().mean()
```

```
0.0
```

```
# After Fill the Marital Status column
df["Marital Status"].to_string()
```

```
'0      Married\n1      Married\n2      Married\n3       Single\n4       Single\n5      Married\n6       Single\n7      Married\n8
Married\n9      Married\n10     Married\n11      Single\n12     Married\n13     Married\n14      Single\n15      Single\n16      Single
\n17     Married\n18      Single\n19      Single\n20     Married\n21      Single\n22      Single\n23     Married\n24      Single\n25
Single\n26      Single\n27     Married\n28     Married\n29      Single\n30     Married\n31     Married\n32      Single\n33      Single
\n34      Single\n35      Single\n36     Married\n37      Single\n38      Single\n39      Single\n40      Single\n41      Single\n42
Married\n43     Married\n44     Married\n45     Married\n46     Married\n47      Single\n48     Married\n49     Married\n50      Single
\n51      Single\n52     Married\n53      Single\n54      Single\n55     Married\n56     Married\n57     Married\n58     Married\n59
```

Filling Data by Bfill (BACKWARD FILL) Method.

```
# Show Gender column.
df["Gender"].to_string()
```

```
'0      Female\n1        Male\n2        Male\n3         NaN\n4        Male\n5      Female\n6        Male\n7        Male\n8        Male
\n9        Male\n10      Female\n11      Female\n12        Male\n13        Male\n14        Male\n15      Female\n16        Male\n17      Female
\n18        Male\n19        Male\n20      Female\n21      Female\n22        Male\n23      Female\n24        Male\n25        Male\n26        Male
\n27      Female\n28        Male\n29      Female\n30      Female\n31        Male\n32      Female\n33        Male\n34        Male\n35      Female
\n36      Female\n37      Female\n38        Male\n39      Female\n40      Female\n41      Female\n42      Female\n43      Female\n44      Female
\n45      Female\n46      Female\n47      Female\n48        Male\n49        Male\n50      Female\n51        Male\n52      Female\n53      Female
```

```
df["Gender"].isnull().mean()
```

```
0.011
```

```
# Filling Gender column with Bfill method (Backward Fill)
df["Gender"].bfill(inplace=True)
```

```
<ipython-input-83-84999d5bb0b1>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df["Gender"].bfill(inplace=True)
```

```
# After fill Gender column
df["Gender"].to_string()
```

```
'0      Female\n1        Male\n2        Male\n3        Male\n4        Male\n5      Female\n6        Male\n7        Male\n8        Male
\n9        Male\n10      Female\n11      Female\n12        Male\n13        Male\n14        Male\n15      Female\n16        Male\n17      Female
\n18        Male\n19        Male\n20      Female\n21      Female\n22        Male\n23      Female\n24        Male\n25        Male\n26        Male
\n27      Female\n28        Male\n29      Female\n30      Female\n31        Male\n32      Female\n33        Male\n34        Male\n35      Female
\n36      Female\n37      Female\n38        Male\n39      Female\n40      Female\n41      Female\n42      Female\n43      Female\n44      Female
\n45      Female\n46      Female\n47      Female\n48        Male\n49        Male\n50      Female\n51        Male\n52      Female\n53      Female
```

```
df["Gender"].isnull().mean()
```

```
0.0
```

Data filling by Ffill (FORWARD FILL) Method

```
# Show Home Owner null values
df["Home Owner"].isnull().mean()
```

```
0.004
```

```
df["Home Owner"].to_string()
```

```
'0      Yes\n1      Yes\n2       No\n3      Yes\n4       No\n5      Yes\n6      NaN\n7      Yes\n8      Yes\n9      Yes\n10      No\n11
No\n12     Yes\n13     Yes\n14      No\n15     Yes\n16      No\n17     Yes\n18     Yes\n19     Yes\n20     Yes\n21     Yes\n22     Yes
\n23      No\n24      No\n25     Yes\n26      No\n27      No\n28     Yes\n29      No\n30     Yes\n31      No\n32      No\n33      No\n3
4      No\n35      No\n36     Yes\n37      No\n38      No\n39     Yes\n40      No\n41     Yes\n42     Yes\n43      No\n44     Yes\n45
Yes\n46     Yes\n47     Yes\n48      No\n49     Yes\n50      No\n51      No\n52     Yes\n53      No\n54      No\n55     Yes\n56     Yes
\n57      No\n58     Yes\n59     Yes\n60      No\n61     Yes\n62     Yes\n63     Yes\n64     Yes\n65     Yes\n66     Yes\n67     Yes\n6
8     Yes\n69      No\n70     Yes\n71      No\n72     Yes\n73      No\n74      No\n75     Yes\n76      No\n77     Yes\n78      No\n79
```

```
# Filling Home Owner column
df["Home Owner"].ffill(inplace=True)
```

```
<ipython-input-88-bc4ad60ae0bd>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df["Home Owner"].ffill(inplace=True)
```

```
df["Home Owner"].isnull().mean()
```

```
0.0
```

```
df["Home Owner"].to_string()
```

```
'0      Yes\n1      Yes\n2       No\n3      Yes\n4       No\n5      Yes\n6      Yes\n7      Yes\n8      Yes\n9      Yes\n10      No\n11
No\n12     Yes\n13     Yes\n14      No\n15     Yes\n16      No\n17     Yes\n18     Yes\n19     Yes\n20     Yes\n21     Yes\n22     Yes
\n23      No\n24      No\n25     Yes\n26      No\n27      No\n28     Yes\n29      No\n30     Yes\n31      No\n32      No\n33      No\n3
4      No\n35      No\n36     Yes\n37      No\n38      No\n39     Yes\n40      No\n41     Yes\n42     Yes\n43      No\n44     Yes\n45
Yes\n46     Yes\n47     Yes\n48      No\n49     Yes\n50      No\n51      No\n52     Yes\n53      No\n54      No\n55     Yes\n56     Yes
\n57      No\n58     Yes\n59     Yes\n60      No\n61     Yes\n62     Yes\n63     Yes\n64     Yes\n65     Yes\n66     Yes\n67     Yes\n6
8     Yes\n69      No\n70     Yes\n71      No\n72     Yes\n73      No\n74      No\n75     Yes\n76      No\n77     Yes\n78      No\n79
```