In [94]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
pd.set_option("display.max_columns", None)
```

In [3]:
```
df = pd.read_csv("Alcohol.csv")
df.head()
```

Out[3]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **0** | 1986 | Western Pacific | Viet Nam | Wine | 0.00 |
| **1** | 1986 | Americas | Uruguay | Other | 0.50 |
| **2** | 1985 | Africa | Cte d'Ivoire | Wine | 1.62 |
| **3** | 1986 | Americas | Colombia | Beer | 4.27 |
| **4** | 1987 | Americas | Saint Kitts and Nevis | Beer | 1.98 |

**Write a Pandas program to select first 2 rows, 2 columns and specific two columns from World alcohol consumption dataset.**

In [4]:
```
print("First two rows")
df.iloc[:2,:]
```

```
First two rows
```

Out[4]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **0** | 1986 | Western Pacific | Viet Nam | Wine | 0.0 |
| **1** | 1986 | Americas | Uruguay | Other | 0.5 |

In [5]:
```
print("First two Columns")
df.iloc[:,:2]
```

```
First two Columns
```

Out[5]:

| | Year | WHO region |
|---|---|---|
| **0** | 1986 | Western Pacific |
| **1** | 1986 | Americas |
| **2** | 1985 | Africa |
| **3** | 1986 | Americas |
| **4** | 1987 | Americas |
| **...** | ... | ... |
| **95** | 1984 | Africa |
| **96** | 1985 | Europe |
| **97** | 1984 | South-East Asia |
| **98** | 1984 | Africa |
| **99** | 1985 | South-East Asia |

100 rows × 2 columns

**Write a Pandas program to find and drop the missing values from World alcohol consumption dataset.**

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
Year              0
WHO region        0
Country           0
Beverage Types    0
Display Value     5
dtype: int64
```

In [7]:
```
# Droping the missing values

df.dropna()
```

Out[7]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **0** | 1986 | Western Pacific | Viet Nam | Wine | 0.00 |
| **1** | 1986 | Americas | Uruguay | Other | 0.50 |
| **2** | 1985 | Africa | Cte d'Ivoire | Wine | 1.62 |
| **3** | 1986 | Americas | Colombia | Beer | 4.27 |
| **4** | 1987 | Americas | Saint Kitts and Nevis | Beer | 1.98 |
| **...** | ... | ... | ... | ... | ... |
| **95** | 1984 | Africa | Niger | Other | 0.00 |
| **96** | 1985 | Europe | Luxembourg | Wine | 7.38 |
| **97** | 1984 | South-East Asia | Indonesia | Wine | 0.00 |
| **98** | 1984 | Africa | Equatorial Guinea | Wine | 0.00 |
| **99** | 1985 | South-East Asia | Democratic People's Republic of Korea | Wine | 0.00 |

95 rows × 5 columns

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Year              0
WHO region        0
Country           0
Beverage Types    0
Display Value     5
dtype: int64
```

**Write a Pandas program to remove the duplicates from 'WHO region' column of World alcohol consumption dataset**

In [9]:
```
df.drop_duplicates("WHO region")
```

Out[9]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **0** | 1986 | Western Pacific | Viet Nam | Wine | 0.00 |
| **1** | 1986 | Americas | Uruguay | Other | 0.50 |
| **2** | 1985 | Africa | Cte d'Ivoire | Wine | 1.62 |
| **13** | 1984 | Eastern Mediterranean | Afghanistan | Other | 0.00 |
| **18** | 1984 | Europe | Norway | Spirits | 1.62 |
| **20** | 1986 | South-East Asia | Myanmar | Wine | 0.00 |

**Write a Pandas program to find out the alcohol consumption details in the year '1987' or**

In [10]:
```
df[(df["Year"]==1987) | (df["Year"]==1989)].head()
```

Out[10]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **4** | 1987 | Americas | Saint Kitts and Nevis | Beer | 1.98 |
| **5** | 1987 | Americas | Guatemala | Other | 0.00 |
| **6** | 1987 | Africa | Mauritius | Wine | 0.13 |
| **10** | 1987 | Africa | Botswana | Wine | 0.20 |
| **11** | 1989 | Americas | Guatemala | Beer | 0.62 |

**Write a Pandas program to find out the alcohol consumption details by the 'Americas' in the year '1985' from the world alcohol consumption dataset.**

In [11]:
```
df[(df["WHO region"]=="Americas") & (df["Year"]==1985)]
```

Out[11]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **35** | 1985 | Americas | Saint Kitts and Nevis | Spirits | 2.24 |

**Write a Pandas program to find out the alcohol consumption details in the year '1986' where WHO region is 'Western Pacific' and country is 'VietNam' from the world alcohol consumption dataset.**

In [12]:
```
df[(df["Year"]==1986) & (df["WHO region"]=="Western Pacific") & (df["Country"]
```

Out[12]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **0** | 1986 | Western Pacific | Viet Nam | Wine | 0.0 |

**Write a Pandas program to find out the alcohol consumption details in the year '1986' or '1989' where WHO region is 'Americas' from the world alcohol consumption dataset.**

In [13]:
```
df[((df["Year"]==1986) | (df["Year"]==1989)) & (df["WHO region"]=="Americas")]
```

Out[13]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **1** | 1986 | Americas | Uruguay | Other | 0.50 |
| **3** | 1986 | Americas | Colombia | Beer | 4.27 |
| **8** | 1986 | Americas | Antigua and Barbuda | Spirits | 1.55 |
| **11** | 1989 | Americas | Guatemala | Beer | 0.62 |
| **21** | 1989 | Americas | Costa Rica | Spirits | 4.51 |
| **47** | 1986 | Americas | Mexico | Other | 0.04 |
| **55** | 1989 | Americas | Suriname | Wine | 0.04 |
| **64** | 1989 | Americas | Bolivia (Plurinational State of) | Beer | 1.26 |
| **74** | 1986 | Americas | Bolivia (Plurinational State of) | Spirits | 2.06 |
| **78** | 1989 | Americas | Jamaica | Other | 0.00 |
| **86** | 1986 | Americas | Bahamas | Wine | 1.83 |

**Write a Pandas program to find out the alcohol consumption details in the year '1985' or '1989' where WHO region is 'Americas' or 'Europe' from the world alcohol consumption dataset.**

In [14]:
```
df[((df["Year"]==1985) | (df["Year"]==1989)) & ((df["WHO region"]=="Americas")
```

Out[14]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **11** | 1989 | Americas | Guatemala | Beer | 0.62 |
| **21** | 1989 | Americas | Costa Rica | Spirits | 4.51 |
| **26** | 1985 | Europe | United Kingdom of Great Britain and Northern I... | Wine | 1.36 |
| **35** | 1985 | Americas | Saint Kitts and Nevis | Spirits | 2.24 |
| **44** | 1985 | Europe | Lithuania | Other | NaN |
| **50** | 1985 | Europe | Switzerland | Other | 0.30 |
| **55** | 1989 | Americas | Suriname | Wine | 0.04 |
| **57** | 1989 | Europe | Croatia | Wine | 5.10 |
| **64** | 1989 | Americas | Bolivia (Plurinational State of) | Beer | 1.26 |
| **78** | 1989 | Americas | Jamaica | Other | 0.00 |
| **79** | 1989 | Europe | Finland | Other | 2.09 |
| **81** | 1985 | Europe | Netherlands | Wine | 2.54 |
| **91** | 1989 | Europe | Bulgaria | Beer | 4.43 |
| **94** | 1985 | Europe | Ukraine | Spirits | 3.06 |
| **96** | 1985 | Europe | Luxembourg | Wine | 7.38 |

**Write a Pandas program to find out the 'WHO region, 'Country', 'Beverage Types' in the year '1986' or '1989' where WHO region is 'Americas' or 'Europe' from the world alcohol consumption dataset.**

In [15]:
```
temp = df[((df["Year"]==1985) | (df["Year"]==1989)) & ((df["WHO region"]=="Ame
temp[["WHO region","Country","Beverage Types"]]
```

Out[15]:

| | WHO region | Country | Beverage Types |
|---|---|---|---|
| **11** | Americas | Guatemala | Beer |
| **21** | Americas | Costa Rica | Spirits |
| **26** | Europe | United Kingdom of Great Britain and Northern I... | Wine |
| **35** | Americas | Saint Kitts and Nevis | Spirits |
| **44** | Europe | Lithuania | Other |
| **50** | Europe | Switzerland | Other |
| **55** | Americas | Suriname | Wine |
| **57** | Europe | Croatia | Wine |
| **64** | Americas | Bolivia (Plurinational State of) | Beer |
| **78** | Americas | Jamaica | Other |
| **79** | Europe | Finland | Other |
| **81** | Europe | Netherlands | Wine |
| **91** | Europe | Bulgaria | Beer |
| **94** | Europe | Ukraine | Spirits |
| **96** | Europe | Luxembourg | Wine |

**Write a Pandas program to find out the records where consumption of beverages per person average >=5 and Beverage Types is Beer from world alcohol consumption dataset.**

In [16]:
```
df[(df["Display Value"]>=5) & (df["Beverage Types"]=="Beer")]
```

Out[16]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **41** | 1986 | Europe | Czech Republic | Beer | 6.82 |

**Write a Pandas program to filter those records where WHO region contains "Ea" substring from world alcohol consumption dataset.**

In [17]:
```
df[df["WHO region"].str.contains("Ea")].head()
```

Out[17]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **13** | 1984 | Eastern Mediterranean | Afghanistan | Other | 0.00 |
| **20** | 1986 | South-East Asia | Myanmar | Wine | 0.00 |
| **25** | 1984 | Eastern Mediterranean | Tunisia | Other | 0.00 |
| **27** | 1984 | Eastern Mediterranean | Bahrain | Beer | 2.22 |
| **36** | 1987 | Eastern Mediterranean | Egypt | Beer | 0.07 |

**Write a Pandas program to filter those records where WHO region matches with multiple values (Africa, Eastern Mediterranean, Europe) from world alcohol consumption dataset.**

In [18]:
```
df[(df["WHO region"]=="Africa") | (df["WHO region"]=="Eastern Mediterranean")
```

Out[18]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **2** | 1985 | Africa | Cte d'Ivoire | Wine | 1.62 |
| **6** | 1987 | Africa | Mauritius | Wine | 0.13 |
| **7** | 1985 | Africa | Angola | Spirits | 0.39 |
| **9** | 1984 | Africa | Nigeria | Other | 6.10 |
| **10** | 1987 | Africa | Botswana | Wine | 0.20 |
| **...** | ... | ... | ... | ... | ... |
| **93** | 1987 | Africa | Madagascar | Other | NaN |
| **94** | 1985 | Europe | Ukraine | Spirits | 3.06 |
| **95** | 1984 | Africa | Niger | Other | 0.00 |
| **96** | 1985 | Europe | Luxembourg | Wine | 7.38 |
| **98** | 1984 | Africa | Equatorial Guinea | Wine | 0.00 |

69 rows × 5 columns

**Write a Pandas program to filter those records which not appears in a given list from world alcohol consumption dataset.**

In [19]:
```
list1 = ["Africa","Eastern Mediterranean","Europe"]

df[~df["WHO region"].isin(list1)].head()
```

Out[19]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **0** | 1986 | Western Pacific | Viet Nam | Wine | 0.00 |
| **1** | 1986 | Americas | Uruguay | Other | 0.50 |
| **3** | 1986 | Americas | Colombia | Beer | 4.27 |
| **4** | 1987 | Americas | Saint Kitts and Nevis | Beer | 1.98 |
| **5** | 1987 | Americas | Guatemala | Other | 0.00 |

**Write a Pandas program to filter all records where the average consumption of beverages per person from .5 to 2.50 in world alcohol consumption dataset.**

In [20]:
```
df[(df["Display Value"] > 0.5) & (df["Display Value"] < 2.50)].head()
```

Out[20]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **2** | 1985 | Africa | Cte d'Ivoire | Wine | 1.62 |
| **4** | 1987 | Americas | Saint Kitts and Nevis | Beer | 1.98 |
| **8** | 1986 | Americas | Antigua and Barbuda | Spirits | 1.55 |
| **11** | 1989 | Americas | Guatemala | Beer | 0.62 |
| **17** | 1989 | Africa | Seychelles | Beer | 2.23 |

**Write a Pandas program to filter rows based on row numbers ended with 0, like 0, 10, 20, 30 from world alcohol consumption dataset.**

In [21]:
```
df.iloc[::10, :]
```

Out[21]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **0** | 1986 | Western Pacific | Viet Nam | Wine | 0.00 |
| **10** | 1987 | Africa | Botswana | Wine | 0.20 |
| **20** | 1986 | South-East Asia | Myanmar | Wine | 0.00 |
| **30** | 1986 | Africa | Sierra Leone | Other | 4.48 |
| **40** | 1987 | Europe | Austria | Spirits | 1.90 |
| **50** | 1985 | Europe | Switzerland | Other | 0.30 |
| **60** | 1987 | Eastern Mediterranean | Iran (Islamic Republic of) | Other | 0.00 |
| **70** | 1986 | Africa | Madagascar | Spirits | 1.02 |
| **80** | 1985 | Africa | Malawi | Other | 0.84 |
| **90** | 1989 | Africa | Malawi | Wine | 0.01 |

**Write a Pandas program to select consecutive columns and also select rows with Index label 0 to 9 with some columns from world alcohol consumption dataset.**

In [22]:
```
df.iloc[:10, ::2]
```

Out[22]:

| | Year | Country | Display Value |
|---|---|---|---|
| **0** | 1986 | Viet Nam | 0.00 |
| **1** | 1986 | Uruguay | 0.50 |
| **2** | 1985 | Cte d'Ivoire | 1.62 |
| **3** | 1986 | Colombia | 4.27 |
| **4** | 1987 | Saint Kitts and Nevis | 1.98 |
| **5** | 1987 | Guatemala | 0.00 |
| **6** | 1987 | Mauritius | 0.13 |
| **7** | 1985 | Angola | 0.39 |
| **8** | 1986 | Antigua and Barbuda | 1.55 |
| **9** | 1984 | Nigeria | 6.10 |

**Write a Pandas program to find which years have all non-zero values and which years have any non-zero values from world alcohol consumption dataset.**

In [23]:
```
df.loc[:, df.all()]
```

Out[23]:

| | Year | WHO region | Country | Beverage Types |
|---|---|---|---|---|
| **0** | 1986 | Western Pacific | Viet Nam | Wine |
| **1** | 1986 | Americas | Uruguay | Other |
| **2** | 1985 | Africa | Cte d'Ivoire | Wine |
| **3** | 1986 | Americas | Colombia | Beer |
| **4** | 1987 | Americas | Saint Kitts and Nevis | Beer |
| **...** | ... | ... | ... | ... |
| **95** | 1984 | Africa | Niger | Other |
| **96** | 1985 | Europe | Luxembourg | Wine |
| **97** | 1984 | South-East Asia | Indonesia | Wine |
| **98** | 1984 | Africa | Equatorial Guinea | Wine |
| **99** | 1985 | South-East Asia | Democratic People's Republic of Korea | Wine |

100 rows × 4 columns

In [24]:
```
df.loc[:, df.any()]
```

Out[24]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **0** | 1986 | Western Pacific | Viet Nam | Wine | 0.00 |
| **1** | 1986 | Americas | Uruguay | Other | 0.50 |
| **2** | 1985 | Africa | Cte d'Ivoire | Wine | 1.62 |
| **3** | 1986 | Americas | Colombia | Beer | 4.27 |
| **4** | 1987 | Americas | Saint Kitts and Nevis | Beer | 1.98 |
| **...** | ... | ... | ... | ... | ... |
| **95** | 1984 | Africa | Niger | Other | 0.00 |
| **96** | 1985 | Europe | Luxembourg | Wine | 7.38 |
| **97** | 1984 | South-East Asia | Indonesia | Wine | 0.00 |
| **98** | 1984 | Africa | Equatorial Guinea | Wine | 0.00 |
| **99** | 1985 | South-East Asia | Democratic People's Republic of Korea | Wine | 0.00 |

100 rows × 5 columns

**Write a Pandas program to filter all columns where all entries present, check which rows and columns has a NaN and finally drop rows with any NaNs from world alcohol consumption dataset.**

In [25]:
```
df.isna().sum()
```

Out[25]:
```
Year              0
WHO region        0
Country           0
Beverage Types    0
Display Value     5
dtype: int64
```

In [26]:
```
df = df.dropna()
df.head()
```

Out[26]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **0** | 1986 | Western Pacific | Viet Nam | Wine | 0.00 |
| **1** | 1986 | Americas | Uruguay | Other | 0.50 |
| **2** | 1985 | Africa | Cte d'Ivoire | Wine | 1.62 |
| **3** | 1986 | Americas | Colombia | Beer | 4.27 |
| **4** | 1987 | Americas | Saint Kitts and Nevis | Beer | 1.98 |

**Write a Pandas program to filter all records starting from the 2nd row, access every 5th row from world alcohol consumption dataset.**

In [27]:
```
df.iloc[1::5, :].head(10)
```

Out[27]:

| | Year | WHO region | Country | Beverage Types | Display Value |
|---|---|---|---|---|---|
| **1** | 1986 | Americas | Uruguay | Other | 0.50 |
| **6** | 1987 | Africa | Mauritius | Wine | 0.13 |
| **11** | 1989 | Americas | Guatemala | Beer | 0.62 |
| **16** | 1984 | Americas | Costa Rica | Wine | 0.06 |
| **21** | 1989 | Americas | Costa Rica | Spirits | 4.51 |
| **27** | 1984 | Eastern Mediterranean | Bahrain | Beer | 2.22 |
| **33** | 1985 | Africa | Mauritania | Other | 0.00 |
| **38** | 1987 | Eastern Mediterranean | Qatar | Other | 0.00 |
| **43** | 1984 | Western Pacific | China | Wine | 0.03 |
| **49** | 1986 | Europe | Malta | Wine | 1.49 |

In [28]:
```
saledf = pd.read_csv("SaleData.csv")
saledf.head()
```

Out[28]:

| | OrderDate | Region | Manager | SalesMan | Item | Units | Unit_price | Sale_amt |
|---|---|---|---|---|---|---|---|---|
| **0** | 1/6/2018 | East | Martha | Alexander | Television | 95 | 1198.0 | 113810.0 |
| **1** | 1/23/2018 | Central | Hermann | Shelli | Home Theater | 50 | 500.0 | 25000.0 |
| **2** | 2/9/2018 | Central | Hermann | Luis | Television | 36 | 1198.0 | 43128.0 |
| **3** | 2/26/2018 | Central | Timothy | David | Cell Phone | 27 | 225.0 | 6075.0 |
| **4** | 3/15/2018 | West | Timothy | Stephen | Television | 56 | 1198.0 | 67088.0 |

In [29]:
```
saledf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43 entries, 0 to 42
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   OrderDate     43 non-null     object 
 1   Region        43 non-null     object 
 2   Manager       43 non-null     object 
 3   SalesMan      43 non-null     object 
 4   Item          43 non-null     object 
 5   Units         43 non-null     int64  
 6    Unit_price   43 non-null     float64
 7   Sale_amt      43 non-null     float64
dtypes: float64(2), int64(1), object(5)
memory usage: 2.8+ KB
```

**Write a Pandas program to create a Pivot table and find the total sale amount region wise, manager wise.**

In [30]:
```
pd.pivot_table(saledf, index=["Region","Manager"], values="Sale_amt", aggfunc=
```

Out[30]:

| | | Sale_amt |
|---|---|---|
| **Region** | **Manager** | |
| **Central** | **Douglas** | 124016.0 |
| | **Hermann** | 365108.5 |
| | **Martha** | 199690.0 |
| | **Timothy** | 140955.0 |
| **East** | **Douglas** | 48204.0 |
| | **Martha** | 272803.0 |
| **West** | **Douglas** | 66836.0 |
| | **Timothy** | 88063.0 |

**Write a Pandas program to create a Pivot table and find the item wise unit sold.**

In [31]:
```
pd.pivot_table(saledf, index="Item", values="Units", aggfunc=np.sum)
```

Out[31]:

| | Units |
|---|---|
| **Item** | |
| **Cell Phone** | 278 |
| **Desk** | 10 |
| **Home Theater** | 722 |
| **Television** | 716 |
| **Video Games** | 395 |

**Write a Pandas program to create a Pivot table and count the manager wise sale and mean value of sale amount.**

In [32]:
```
pd.pivot_table(saledf, index="Manager", values="Sale_amt", aggfunc=[len, np.me
```

Out[32]:

| | len | mean |
|---|---|---|
| | **Sale_amt** | **Sale_amt** |
| **Manager** | | |
| **Douglas** | 8 | 29882.000000 |
| **Hermann** | 12 | 30425.708333 |
| **Martha** | 14 | 33749.500000 |
| **Timothy** | 9 | 25446.444444 |

**Write a Pandas program to create a Pivot table and find manager wise, salesman wise total sale and also display the sum of all sale amount at the bottom.**

In [33]:
```
pd.pivot_table(saledf, index=["Manager","SalesMan"], values="Sale_amt", aggfur
```

Out[33]:

| | | Sale_amt |
|---|---|---|
| **Manager** | **SalesMan** | |
| **Douglas** | **John** | 124016.0 |
| | **Karen** | 48204.0 |
| | **Michael** | 66836.0 |
| **Hermann** | **Luis** | 206373.0 |
| | **Shelli** | 33698.0 |
| | **Sigal** | 125037.5 |
| **Martha** | **Alexander** | 236703.0 |
| | **Diana** | 36100.0 |
| **Timothy** | **Steven** | 199690.0 |
| | **David** | 140955.0 |
| | **Stephen** | 88063.0 |
| **All** | | 1305675.5 |

**Write a Pandas program to create a Pivot table and find the total sale amount region wise, manager wise, sales man wise where Manager = "Douglas".**

In [34]:
```
temp = pd.pivot_table(saledf, index=["Region","Manager","SalesMan"], values="S
temp.query('Manager==["Douglas"]')
```

Out[34]:

| | | | Sale_amt |
|---|---|---|---|
| **Region** | **Manager** | **SalesMan** | |
| **Central** | **Douglas** | **John** | 124016.0 |
| **East** | **Douglas** | **Karen** | 48204.0 |
| **West** | **Douglas** | **Michael** | 66836.0 |

**Write a Pandas program to create a Pivot table and find the region wise Television and Home Theater sold.**

In [35]:
```
temp = pd.pivot_table(saledf, index=["Region","Item"], values="Units", aggfunc
temp.query('Item==["Television","Home Theater"]')
```

Out[35]:

| | | Units |
|---|---|---|
| **Region** | **Item** | |
| **Central** | **Home Theater** | 424 |
| | **Television** | 498 |
| **East** | **Home Theater** | 234 |
| | **Television** | 130 |
| **West** | **Home Theater** | 64 |
| | **Television** | 88 |

**Write a Pandas program to create a Pivot table and find the maximum and minimum sale value of the items.**

In [36]:
```
pd.pivot_table(saledf, index="Item", values="Sale_amt", aggfunc=[np.max, np.mi
```

Out[36]:

| | amax | amin |
|---|---|---|
| | **Sale_amt** | **Sale_amt** |
| **Item** | | |
| **Cell Phone** | 21600.0 | 3375.0 |
| **Desk** | 625.0 | 250.0 |
| **Home Theater** | 47000.0 | 2000.0 |
| **Television** | 113810.0 | 8386.0 |
| **Video Games** | 5616.0 | 936.0 |

In [37]:
```
titanic = pd.read_csv("titanic.csv")
titanic.head()
```

Out[37]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True |

**Write a Pandas program to create a Pivot table and find survival rate by gender on various classes**

In [38]:
```
titanic.pivot_table("survived", index="sex", columns="class")
```

Out[38]:

| sex | First | Second | Third |
|---|---|---|---|
| female | 0.968085 | 0.921053 | 0.500000 |
| male | 0.368852 | 0.157407 | 0.135447 |

**Write a Pandas program to create a Pivot table and find survival rate by gender.**

In [39]:
```
titanic.groupby("sex")["survived"].mean()
```

Out[39]:
```
sex
female    0.742038
male      0.188908
Name: survived, dtype: float64
```

**Write a Pandas program to create a Pivot table and find survival rate by gender, age of the different categories of various classes. Add the fare as a dimension of columns and partition fare column into 2 categories based on the values present in fare columns.**

In [40]:
```
fare = pd.qcut(titanic['fare'], 2)
age = pd.cut(titanic['age'], [0, 10, 30, 60, 80])
titanic.pivot_table('survived', index=['sex', age], columns=[fare, 'pclass'])
```

Out[40]:

| | fare | (-0.001, 14.454] | | | (14.454, 512.329] | | |
|---|---|---|---|---|---|---|---|
| | pclass | 1 | 2 | 3 | 1 | 2 | 3 |
| sex | age | | | | | | |
| female | (0, 10] | NaN | NaN | 0.800000 | 0.000000 | 1.000000 | 0.411765 |
| | (10, 30] | NaN | 0.933333 | 0.568182 | 0.970588 | 0.904762 | 0.307692 |
| | (30, 60] | NaN | 0.846154 | 0.142857 | 0.979167 | 0.941176 | 0.333333 |
| | (60, 80] | NaN | NaN | 1.000000 | 1.000000 | NaN | NaN |
| male | (0, 10] | NaN | NaN | 1.000000 | 1.000000 | 1.000000 | 0.263158 |
| | (10, 30] | NaN | 0.034483 | 0.140625 | 0.458333 | 0.000000 | 0.130435 |
| | (30, 60] | 0.0 | 0.130435 | 0.109375 | 0.440678 | 0.047619 | 0.166667 |
| | (60, 80] | NaN | 0.333333 | 0.000000 | 0.083333 | NaN | NaN |

**Write a Pandas program to create a Pivot table and find number of adult male, adult female and children.**

In [41]:
```
titanic.groupby("who")["who"].count()
```

Out[41]:
```
who
child     83
man      537
woman    271
Name: who, dtype: int64
```

In [42]:
```
df = pd.DataFrame({
'ord_no':[70001,np.nan,70002,70004,np.nan,70005,np.nan,70010,70003,70012,np.na
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,1983.43,2480.4,250.45,
'ord_date': ['2012-10-05','2012-09-10',np.nan,'2012-08-17','2012-09-10','2012-
'customer_id':[3002,3001,3001,3003,3002,3001,3001,3004,3003,3002,3001,3001],
'salesman_id':[5002,5003,5001,np.nan,5002,5001,5001,np.nan,5003,5002,5003,np.n

df
```

Out[42]:

| | ord_no | purch_amt | ord_date | customer_id | salesman_id |
|---|---|---|---|---|---|
| **0** | 70001.0 | 150.50 | 2012-10-05 | 3002 | 5002.0 |
| **1** | NaN | 270.65 | 2012-09-10 | 3001 | 5003.0 |
| **2** | 70002.0 | 65.26 | NaN | 3001 | 5001.0 |
| **3** | 70004.0 | 110.50 | 2012-08-17 | 3003 | NaN |
| **4** | NaN | 948.50 | 2012-09-10 | 3002 | 5002.0 |
| **5** | 70005.0 | 2400.60 | 2012-07-27 | 3001 | 5001.0 |
| **6** | NaN | 5760.00 | 2012-09-10 | 3001 | 5001.0 |
| **7** | 70010.0 | 1983.43 | 2012-10-10 | 3004 | NaN |
| **8** | 70003.0 | 2480.40 | 2012-10-10 | 3003 | 5003.0 |
| **9** | 70012.0 | 250.45 | 2012-06-27 | 3002 | 5002.0 |
| **10** | NaN | 75.29 | 2012-08-17 | 3001 | 5003.0 |
| **11** | 70013.0 | 3045.60 | 2012-04-25 | 3001 | NaN |

**Write a Pandas program to identify the column(s) of a given DataFrame which have at least one missing value.**

In [43]:
```
df.isna().sum()
```

Out[43]:
```
ord_no         4
purch_amt      0
ord_date       1
customer_id    0
salesman_id    3
dtype: int64
```

**Write a Pandas program to find and replace the missing values in a given DataFrame which do not have any valuable information.**

In [44]:
```
df = pd.DataFrame({
'ord_no':[70001,np.nan,70002,70004,np.nan,70005,"--",70010,70003,70012,np.nan,
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,"?",12.43,2480.4,250.4
'ord_date': ['?','2012-09-10',np.nan,'2012-08-17','2012-09-10','2012-07-27','2
'customer_id':[3002,3001,3001,3003,3002,3001,3001,3004,"--",3002,3001,3001],
'salesman_id':[5002,5003,"?",5001,np.nan,5002,5001,"?",5003,5002,5003,"--"]})

df
```

Out[44]:

| | ord_no | purch_amt | ord_date | customer_id | salesman_id |
|---|---|---|---|---|---|
| **0** | 70001 | 150.5 | ? | 3002 | 5002 |
| **1** | NaN | 270.65 | 2012-09-10 | 3001 | 5003 |
| **2** | 70002 | 65.26 | NaN | 3001 | ? |
| **3** | 70004 | 110.5 | 2012-08-17 | 3003 | 5001 |
| **4** | NaN | 948.5 | 2012-09-10 | 3002 | NaN |
| **5** | 70005 | 2400.6 | 2012-07-27 | 3001 | 5002 |
| **6** | -- | 5760 | 2012-09-10 | 3001 | 5001 |
| **7** | 70010 | ? | 2012-10-10 | 3004 | ? |
| **8** | 70003 | 12.43 | 2012-10-10 | -- | 5003 |
| **9** | 70012 | 2480.4 | 2012-06-27 | 3002 | 5002 |
| **10** | NaN | 250.45 | 2012-08-17 | 3001 | 5003 |
| **11** | 70013 | 3045.6 | 2012-04-25 | 3001 | -- |

In [45]:
```
df = df.replace({'--':np.nan, '?':np.nan})
df
```

Out[45]:

| | ord_no | purch_amt | ord_date | customer_id | salesman_id |
|---|---|---|---|---|---|
| **0** | 70001.0 | 150.50 | NaN | 3002.0 | 5002.0 |
| **1** | NaN | 270.65 | 2012-09-10 | 3001.0 | 5003.0 |
| **2** | 70002.0 | 65.26 | NaN | 3001.0 | NaN |
| **3** | 70004.0 | 110.50 | 2012-08-17 | 3003.0 | 5001.0 |
| **4** | NaN | 948.50 | 2012-09-10 | 3002.0 | NaN |
| **5** | 70005.0 | 2400.60 | 2012-07-27 | 3001.0 | 5002.0 |
| **6** | NaN | 5760.00 | 2012-09-10 | 3001.0 | 5001.0 |
| **7** | 70010.0 | NaN | 2012-10-10 | 3004.0 | NaN |
| **8** | 70003.0 | 12.43 | 2012-10-10 | NaN | 5003.0 |
| **9** | 70012.0 | 2480.40 | 2012-06-27 | 3002.0 | 5002.0 |
| **10** | NaN | 250.45 | 2012-08-17 | 3001.0 | 5003.0 |
| **11** | 70013.0 | 3045.60 | 2012-04-25 | 3001.0 | NaN |

**Write a Pandas program to drop the rows where at least one element is missing in a given DataFrame.**

In [46]:
```
df = pd.DataFrame({
'ord_no':[70001,np.nan,70002,70004,np.nan,70005,np.nan,70010,70003,70012,np.na
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,1983.43,2480.4,250.45,
'ord_date': ['2012-10-05','2012-09-10',np.nan,'2012-08-17','2012-09-10','2012-
'customer_id':[3002,3001,3001,3003,3002,3001,3001,3004,3003,3002,3001,3001],
'salesman_id':[5002,5003,5001,np.nan,5002,5001,5001,np.nan,5003,5002,5003,np.n

df
```

Out[46]:

| | ord_no | purch_amt | ord_date | customer_id | salesman_id |
|---|---|---|---|---|---|
| **0** | 70001.0 | 150.50 | 2012-10-05 | 3002 | 5002.0 |
| **1** | NaN | 270.65 | 2012-09-10 | 3001 | 5003.0 |
| **2** | 70002.0 | 65.26 | NaN | 3001 | 5001.0 |
| **3** | 70004.0 | 110.50 | 2012-08-17 | 3003 | NaN |
| **4** | NaN | 948.50 | 2012-09-10 | 3002 | 5002.0 |
| **5** | 70005.0 | 2400.60 | 2012-07-27 | 3001 | 5001.0 |
| **6** | NaN | 5760.00 | 2012-09-10 | 3001 | 5001.0 |
| **7** | 70010.0 | 1983.43 | 2012-10-10 | 3004 | NaN |
| **8** | 70003.0 | 2480.40 | 2012-10-10 | 3003 | 5003.0 |
| **9** | 70012.0 | 250.45 | 2012-06-27 | 3002 | 5002.0 |
| **10** | NaN | 75.29 | 2012-08-17 | 3001 | 5003.0 |
| **11** | 70013.0 | 3045.60 | 2012-04-25 | 3001 | NaN |

In [47]:
```
df.dropna(axis="columns")
```

Out[47]:

| | purch_amt | customer_id |
|---|---|---|
| **0** | 150.50 | 3002 |
| **1** | 270.65 | 3001 |
| **2** | 65.26 | 3001 |
| **3** | 110.50 | 3003 |
| **4** | 948.50 | 3002 |
| **5** | 2400.60 | 3001 |
| **6** | 5760.00 | 3001 |
| **7** | 1983.43 | 3004 |
| **8** | 2480.40 | 3003 |
| **9** | 250.45 | 3002 |
| **10** | 75.29 | 3001 |
| **11** | 3045.60 | 3001 |

**Write a Pandas program to read a dataset from diamonds DataFrame and modify the default columns values and print the first 6 rows**

In [48]:
```
diamond = pd.read_csv("diamond.csv")
diamond.head(6)
```

Out[48]:

| | carat | cut | color | clarity | depth | table | price | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.23 | Ideal | E | SI2 | 61.5 | 55.0 | 326 | 3.95 | 3.98 | 2.43 |
| **1** | 0.21 | Premium | E | SI1 | 59.8 | 61.0 | 326 | 3.89 | 3.84 | 2.31 |
| **2** | 0.23 | Good | E | VS1 | 56.9 | 65.0 | 327 | 4.05 | 4.07 | 2.31 |
| **3** | 0.29 | Premium | I | VS2 | 62.4 | 58.0 | 334 | 4.20 | 4.23 | 2.63 |
| **4** | 0.31 | Good | J | SI2 | 63.3 | 58.0 | 335 | 4.34 | 4.35 | 2.75 |
| **5** | 0.24 | Very Good | J | VVS2 | 62.8 | 57.0 | 336 | 3.94 | 3.96 | 2.48 |

**Write a Pandas program to create a new 'Quality -color' Series (use bracket notation to define the Series name) of the diamonds DataFrame.**

In [49]:
```
diamond["Quality-color"] = diamond.cut + ',' + diamond.color
diamond.head()
```

Out[49]:

| | carat | cut | color | clarity | depth | table | price | x | y | z | Quality-color |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.23 | Ideal | E | SI2 | 61.5 | 55.0 | 326 | 3.95 | 3.98 | 2.43 | Ideal,E |
| **1** | 0.21 | Premium | E | SI1 | 59.8 | 61.0 | 326 | 3.89 | 3.84 | 2.31 | Premium,E |
| **2** | 0.23 | Good | E | VS1 | 56.9 | 65.0 | 327 | 4.05 | 4.07 | 2.31 | Good,E |
| **3** | 0.29 | Premium | I | VS2 | 62.4 | 58.0 | 334 | 4.20 | 4.23 | 2.63 | Premium,I |
| **4** | 0.31 | Good | J | SI2 | 63.3 | 58.0 | 335 | 4.34 | 4.35 | 2.75 | Good,J |

**Write a Pandas program to find the number of rows and columns and data type of each column of diamonds Dataframe.**

In [50]:
```
diamond.info() #Data Type
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   carat          53940 non-null  float64
 1   cut            53940 non-null  object 
 2   color          53940 non-null  object 
 3   clarity        53940 non-null  object 
 4   depth          53940 non-null  float64
 5   table          53940 non-null  float64
 6   price          53940 non-null  int64  
 7   x              53940 non-null  float64
 8   y              53940 non-null  float64
 9   z              53940 non-null  float64
 10  Quality-color  53940 non-null  object 
dtypes: float64(6), int64(1), object(4)
memory usage: 4.5+ MB
```

In [51]:
```
diamond.shape #Rows and columns
```

Out[51]: (53940, 11)

**Write a Pandas program to sort the entire diamonds DataFrame by the 'carat' Series in ascending and descending order.**

In [52]:
```
diamond.sort_values("carat") #Ascending
```

Out[52]:

| | carat | cut | color | clarity | depth | table | price | x | y | z | Quality-color |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **31593** | 0.20 | Premium | E | VS2 | 61.1 | 59.0 | 367 | 3.81 | 3.78 | 2.32 | Premium,E |
| **31597** | 0.20 | Ideal | D | VS2 | 61.5 | 57.0 | 367 | 3.81 | 3.77 | 2.33 | Ideal,D |
| **31596** | 0.20 | Premium | F | VS2 | 62.6 | 59.0 | 367 | 3.73 | 3.71 | 2.33 | Premium,F |
| **31595** | 0.20 | Ideal | E | VS2 | 59.7 | 55.0 | 367 | 3.86 | 3.84 | 2.30 | Ideal,E |
| **31594** | 0.20 | Premium | E | VS2 | 59.7 | 62.0 | 367 | 3.84 | 3.80 | 2.28 | Premium,E |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **25999** | 4.01 | Premium | J | I1 | 62.5 | 62.0 | 15223 | 10.02 | 9.94 | 6.24 | Premium,J |
| **25998** | 4.01 | Premium | I | I1 | 61.0 | 61.0 | 15223 | 10.14 | 10.10 | 6.17 | Premium,I |
| **27130** | 4.13 | Fair | H | I1 | 64.8 | 61.0 | 17329 | 10.00 | 9.85 | 6.43 | Fair,H |
| **27630** | 4.50 | Fair | J | I1 | 65.8 | 58.0 | 18531 | 10.23 | 10.16 | 6.72 | Fair,J |
| **27415** | 5.01 | Fair | J | I1 | 65.5 | 59.0 | 18018 | 10.74 | 10.54 | 6.98 | Fair,J |

53940 rows × 11 columns

In [53]:
```
diamond.sort_values("carat", ascending=False) #Descending
```

Out[53]:

| | carat | cut | color | clarity | depth | table | price | x | y | z | Quality-color |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **27415** | 5.01 | Fair | J | I1 | 65.5 | 59.0 | 18018 | 10.74 | 10.54 | 6.98 | Fair,J |
| **27630** | 4.50 | Fair | J | I1 | 65.8 | 58.0 | 18531 | 10.23 | 10.16 | 6.72 | Fair,J |
| **27130** | 4.13 | Fair | H | I1 | 64.8 | 61.0 | 17329 | 10.00 | 9.85 | 6.43 | Fair,H |
| **25999** | 4.01 | Premium | J | I1 | 62.5 | 62.0 | 15223 | 10.02 | 9.94 | 6.24 | Premium,J |
| **25998** | 4.01 | Premium | I | I1 | 61.0 | 61.0 | 15223 | 10.14 | 10.10 | 6.17 | Premium,I |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **31592** | 0.20 | Premium | E | VS2 | 59.0 | 60.0 | 367 | 3.81 | 3.78 | 2.24 | Premium,E |
| **31591** | 0.20 | Premium | E | VS2 | 59.8 | 62.0 | 367 | 3.79 | 3.77 | 2.26 | Premium,E |
| **31601** | 0.20 | Premium | D | VS2 | 61.7 | 60.0 | 367 | 3.77 | 3.72 | 2.31 | Premium,D |
| **14** | 0.20 | Premium | E | SI2 | 60.2 | 62.0 | 345 | 3.79 | 3.75 | 2.27 | Premium,E |
| **31596** | 0.20 | Premium | F | VS2 | 62.6 | 59.0 | 367 | 3.73 | 3.71 | 2.33 | Premium,F |

53940 rows × 11 columns

**Write a Pandas program to find the details of the diamonds where length>5, width>5 and depth>5**

In [54]:
```
diamond[(diamond.x > 5) & (diamond.y > 5) & (diamond.z > 5)]
```

Out[54]:

| | carat | cut | color | clarity | depth | table | price | x | y | z | Quality-color |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **11778** | 1.83 | Fair | J | I1 | 70.0 | 58.0 | 5083 | 7.34 | 7.28 | 5.12 | Fair,J |
| **13002** | 2.14 | Fair | J | I1 | 69.4 | 57.0 | 5405 | 7.74 | 7.70 | 5.36 | Fair,J |
| **13118** | 2.15 | Fair | J | I1 | 65.5 | 57.0 | 5430 | 8.01 | 7.95 | 5.23 | Fair,J |
| **13562** | 1.96 | Fair | F | I1 | 66.6 | 60.0 | 5554 | 7.59 | 7.56 | 5.04 | Fair,F |
| **13757** | 2.22 | Fair | J | I1 | 66.7 | 56.0 | 5607 | 8.04 | 8.02 | 5.36 | Fair,J |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **27748** | 2.00 | Very Good | G | SI1 | 63.5 | 56.0 | 18818 | 7.90 | 7.97 | 5.04 | Very Good,G |
| **27749** | 2.29 | Premium | I | VS2 | 60.8 | 60.0 | 18823 | 8.50 | 8.47 | 5.16 | Premium,I |
| **48410** | 0.51 | Very Good | E | VS1 | 61.8 | 54.7 | 1970 | 5.12 | 5.15 | 31.80 | Very Good,E |
| **49189** | 0.51 | Ideal | E | VS1 | 61.8 | 55.0 | 2075 | 5.15 | 31.80 | 5.12 | Ideal,E |
| **49905** | 0.50 | Very Good | G | VVS1 | 63.7 | 58.0 | 2180 | 5.01 | 5.04 | 5.06 | Very Good,G |

1457 rows × 11 columns

In [55]:
```
auto = pd.read_csv("auto.csv")
auto
```

Out[55]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | India |
| **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | India |
| **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | India |
| **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | India |
| **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | India |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **393** | 27.0 | 4 | 140.0 | 86.0 | 2790 | 15.6 | 82 | India |
| **394** | 44.0 | 4 | 97.0 | 52.0 | 2130 | 24.6 | 82 | USA |
| **395** | 32.0 | 4 | 135.0 | 84.0 | 2295 | 11.6 | 82 | India |
| **396** | 28.0 | 4 | 120.0 | 79.0 | 2625 | 18.6 | 82 | India |
| **397** | 31.0 | 4 | 119.0 | 82.0 | 2720 | 19.4 | 82 | India |

398 rows × 8 columns

In [56]:
```
# Rename Columns

auto.columns = ["MPG", "Cylinders", "Displacement", "Horsepower", "Weight", "A
auto.head()
```

Out[56]:

| | MPG | Cylinders | Displacement | Horsepower | Weight | Acceleration | Model Year | Origin |
|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | India |
| **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | India |
| **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | India |
| **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | India |
| **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | India |

In [57]:
```
auto.shape
```

Out[57]:
```
(398, 8)
```

In [58]:
```
auto.columns
```

Out[58]:
```
Index(['MPG', 'Cylinders', 'Displacement', 'Horsepower', 'Weight',
       'Acceleration', 'Model Year', 'Origin'],
      dtype='object')
```

In [59]:
```
list(auto.columns)
```

Out[59]:
```
['MPG',
 'Cylinders',
 'Displacement',
 'Horsepower',
 'Weight',
 'Acceleration',
 'Model Year',
 'Origin']
```

In [60]:
```
auto.index
```

Out[60]:
```
RangeIndex(start=0, stop=398, step=1)
```

In [61]:
```
auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   MPG           398 non-null    float64
 1   Cylinders     398 non-null    int64  
 2   Displacement  398 non-null    float64
 3   Horsepower    392 non-null    float64
 4   Weight        398 non-null    int64  
 5   Acceleration  398 non-null    float64
 6   Model Year    398 non-null    int64  
 7   Origin        398 non-null    object 
dtypes: float64(4), int64(3), object(1)
memory usage: 25.0+ KB
```

**How the Numerical and Categorical columns are distributed in the data.**

In [62]:
```
auto.describe()
```

Out[62]:

| | MPG | Cylinders | Displacement | Horsepower | Weight | Acceleration | Model Yea |
|---|---|---|---|---|---|---|---|
| **count** | 398.000000 | 398.000000 | 398.000000 | 392.000000 | 398.000000 | 398.000000 | 398.00000 |
| **mean** | 23.514573 | 5.454774 | 193.425879 | 104.469388 | 2970.424623 | 15.568090 | 76.01005 |
| **std** | 7.815984 | 1.701004 | 104.269838 | 38.491160 | 846.841774 | 2.757689 | 3.69762 |
| **min** | 9.000000 | 3.000000 | 68.000000 | 46.000000 | 1613.000000 | 8.000000 | 70.00000 |
| **25%** | 17.500000 | 4.000000 | 104.250000 | 75.000000 | 2223.750000 | 13.825000 | 73.00000 |
| **50%** | 23.000000 | 4.000000 | 148.500000 | 93.500000 | 2803.500000 | 15.500000 | 76.00000 |
| **75%** | 29.000000 | 8.000000 | 262.000000 | 126.000000 | 3608.000000 | 17.175000 | 79.00000 |
| **max** | 46.600000 | 8.000000 | 455.000000 | 230.000000 | 5140.000000 | 24.800000 | 82.00000 |

In [63]:
```
auto.describe(include="object")
```

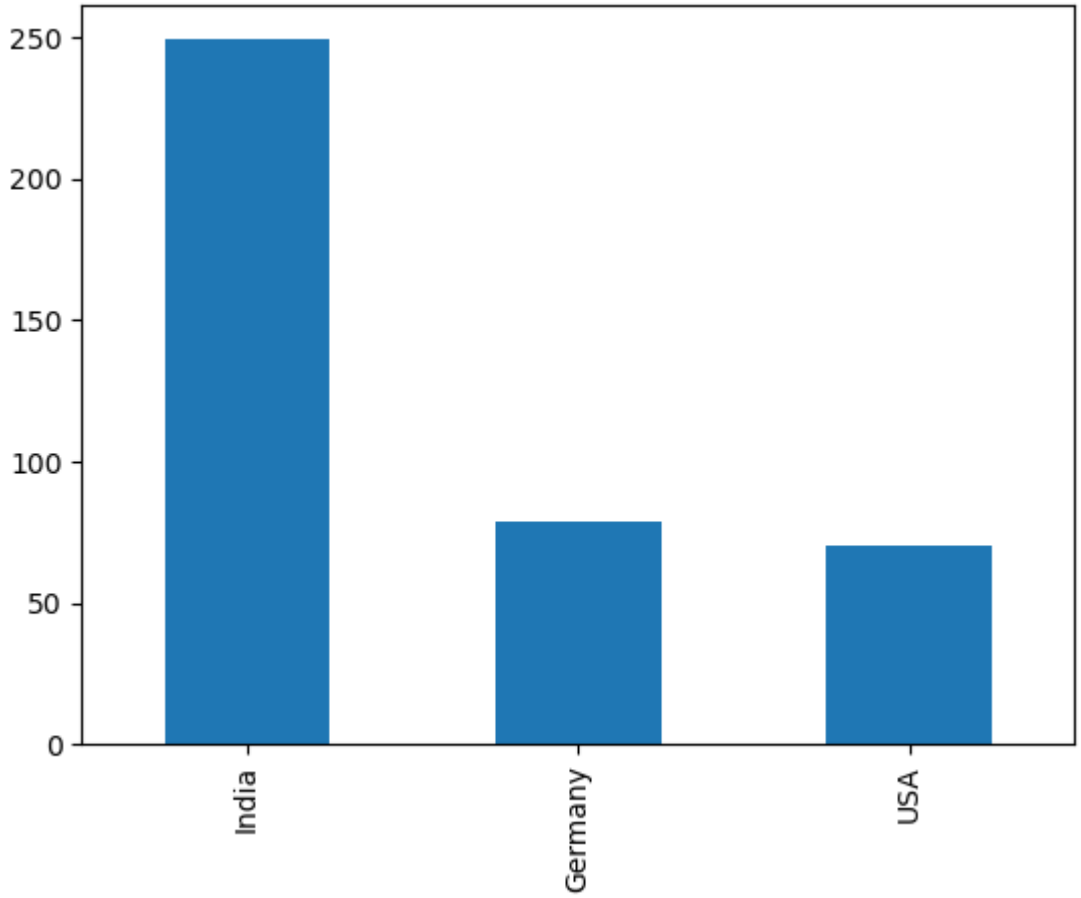
Out[63]:

| | Origin |
|---|---|
| **count** | 398 |
| **unique** | 3 |
| **top** | India |
| **freq** | 249 |

In [64]:
```
auto.Origin.value_counts()
```

Out[64]:
```
India      249
Germany     79
USA         70
Name: Origin, dtype: int64
```

In [65]:
```
auto.Origin.value_counts().plot.bar()
plt.show()
```



In [66]:
```
auto.head()
```

Out[66]:

| | MPG | Cylinders | Displacement | Horsepower | Weight | Acceleration | Model Year | Origin |
|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | India |
| **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | India |
| **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | India |
| **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | India |
| **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | India |

**Add column and display 1. Displacement per unit power 2. Weight per cylinder 3. Acceleration per unit power.**

In [67]:
```
auto["disp_per_unit_power"] = auto.Displacement/auto.Horsepower
auto.head()
```

Out[67]:

| | MPG | Cylinders | Displacement | Horsepower | Weight | Acceleration | Model Year | Origin | disp_per_u |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | India | |
| **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | India | |
| **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | India | |
| **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | India | |
| **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | India | |

In [68]:
```
auto["Weignt_per_cylinder"] = auto.Weight/auto.Cylinders
auto.head()
```

Out[68]:

| | MPG | Cylinders | Displacement | Horsepower | Weight | Acceleration | Model Year | Origin | disp_per_u |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | India | |
| **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | India | |
| **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | India | |
| **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | India | |
| **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | India | |

In [69]:
```
auto["acc_per_unit"] = auto.Acceleration/auto.Horsepower
auto.head()
```

Out[69]:

| | MPG | Cylinders | Displacement | Horsepower | Weight | Acceleration | Model Year | Origin | disp_per_u |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130.0 | 3504 | 12.0 | 70 | India | |
| **1** | 15.0 | 8 | 350.0 | 165.0 | 3693 | 11.5 | 70 | India | |
| **2** | 18.0 | 8 | 318.0 | 150.0 | 3436 | 11.0 | 70 | India | |
| **3** | 16.0 | 8 | 304.0 | 150.0 | 3433 | 12.0 | 70 | India | |
| **4** | 17.0 | 8 | 302.0 | 140.0 | 3449 | 10.5 | 70 | India | |

In [223]:
```
marketing = pd.read_csv("marketing_data.csv")
marketing.head()
```

Out[223]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Re |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | 84835.0 | 0 | 0 | 6/16/2014 | |
| **1** | 1 | 1961 | Graduation | Single | 57091.0 | 0 | 0 | 6/15/2014 | |
| **2** | 10476 | 1958 | Graduation | Married | 67267.0 | 0 | 1 | 5/13/2014 | |
| **3** | 1386 | 1967 | Graduation | Together | 32474.0 | 1 | 1 | 5/11/2014 | |
| **4** | 5371 | 1989 | Graduation | Single | 21474.0 | 1 | 0 | 4/8/2014 | |

In [71]:
```
marketing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 28 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   ID                   2240 non-null   int64 
 1   Year_Birth           2240 non-null   int64 
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
 4    Income              2216 non-null   object
 5   Kidhome              2240 non-null   int64 
 6   Teenhome             2240 non-null   int64 
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64 
 9   MntWines             2240 non-null   int64 
 10  MntFruits            2240 non-null   int64 
 11  MntMeatProducts      2240 non-null   int64 
 12  MntFishProducts      2240 non-null   int64 
 13  MntSweetProducts     2240 non-null   int64 
```

In [72]:
```
marketing.set_index("ID", inplace=True)
```

In [73]:
```
temp = marketing[["MntWines","MntFruits","MntMeatProducts","MntFishProducts","
temp
```

Out[73]:

| ID | MntWines | MntFruits | MntMeatProducts | MntFishProducts | MntSweetProducts | MntGold |
|---|---|---|---|---|---|---|
| **1826** | 189 | 104 | 379 | 111 | 189 | |
| **1** | 464 | 5 | 64 | 7 | 0 | |
| **10476** | 134 | 11 | 59 | 15 | 2 | |
| **1386** | 10 | 0 | 1 | 0 | 0 | |
| **5371** | 6 | 16 | 24 | 11 | 0 | |
| **...** | ... | ... | ... | ... | ... | |
| **10142** | 372 | 18 | 126 | 47 | 48 | |
| **5263** | 5 | 10 | 13 | 3 | 8 | |
| **22** | 185 | 2 | 88 | 15 | 5 | |
| **528** | 267 | 38 | 701 | 149 | 165 | |

In [74]:
```
# Display data from rows 7446 to 2114

temp.loc[7446:2114]
```

Out[74]:

| ID | MntWines | MntFruits | MntMeatProducts | MntFishProducts | MntSweetProducts | MntGoldPro |
|---|---|---|---|---|---|---|
| **7446** | 520 | 42 | 98 | 0 | 42 | |
| **87** | 0 | 7 | 5 | 26 | 2 | |
| **10477** | 71 | 0 | 18 | 0 | 0 | |
| **6072** | 918 | 57 | 842 | 99 | 38 | 1 |
| **2518** | 5 | 4 | 5 | 4 | 2 | |
| **...** | ... | ... | ... | ... | ... | |
| **1802** | 412 | 5 | 119 | 38 | 29 | |
| **1162** | 124 | 83 | 267 | 85 | 59 | |
| **10643** | 124 | 83 | 267 | 85 | 59 | |
| **11112** | 736 | 114 | 279 | 82 | 76 | 1 |
| **2114** | 1006 | 22 | 115 | 59 | 68 | |

155 rows × 6 columns

In [75]:
```
marketing.reset_index(inplace=True)
```

In [76]:
```
# Filter out every alternate rows from index 50 to 300 for the last 9 columns

marketing.iloc[50:301:2, -9:]
```

Out[76]:

| | NumWebVisitsMonth | AcceptedCmp3 | AcceptedCmp4 | AcceptedCmp5 | AcceptedCmp1 | Acce |
|---|---|---|---|---|---|---|
| **50** | 9 | 0 | 0 | 0 | 0 | |
| **52** | 8 | 0 | 0 | 0 | 0 | |
| **54** | 8 | 0 | 0 | 0 | 0 | |
| **56** | 7 | 0 | 0 | 0 | 0 | |
| **58** | 0 | 0 | 0 | 0 | 0 | |
| **...** | ... | ... | ... | ... | ... | |
| **292** | 4 | 1 | 0 | 0 | 0 | |
| **294** | 7 | 0 | 0 | 0 | 0 | |
| **296** | 7 | 0 | 0 | 0 | 0 | |
| **298** | 6 | 0 | 1 | 0 | 0 | |
| **300** | 1 | 0 | 1 | 1 | 0 | |

126 rows × 9 columns

In [77]:
```
land = pd.read_csv("LandSlide.csv")
land.head()
```

Out[77]:

| | id | date | time | country_name | state/province | population | landslide_type | trigger | fata |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 34 | 3/2/2007 | Night | United States | Virginia | 16000 | Landslide | Rain | |
| **1** | 42 | 3/22/2007 | NaN | United States | Ohio | 17288 | Landslide | Rain | |
| **2** | 56 | 4/6/2007 | NaN | United States | Pennsylvania | 15930 | Landslide | Rain | |
| **3** | 59 | 4/14/2007 | NaN | Canada | Quebec | 42786 | Riverbank collapse | Rain | |
| **4** | 61 | 4/15/2007 | NaN | United States | Kentucky | 6903 | Landslide | Downpour | |

In [78]:
```
# Checking the missing values

land.isna().sum()
```

Out[78]:
```
id                   0
date                 3
time              1065
country_name         0
state/province       1
population           0
landslide_type       1
trigger              2
fatalities         247
dtype: int64
```

In [79]:
```
land = land[~land.date.isna()] #Droping null values in date column
```

In [80]:
```
land.isna().sum()
```

Out[80]:
```
id                   0
date                 0
time              1065
country_name         0
state/province       1
population           0
landslide_type       1
trigger              2
fatalities         247
dtype: int64
```

In [81]:
```
land.time.value_counts()
```

Out[81]:
```
Night            97
Morning          87
Afternoon        58
Early morning    36
3:00:00          12
                 ..
3:20:00           1
1:13              1
15:32             1
11:50:00          1
21:06             1
Name: time, Length: 158, dtype: int64
```

In [82]:
```
land.time.fillna("Not Known", inplace=True)
land.isnull().sum()
```

Out[82]:
```
id                  0
date                0
time                0
country_name        0
state/province      1
population          0
landslide_type      1
trigger             2
fatalities        247
dtype: int64
```

In [83]:
```
land.fatalities.fillna(land.fatalities.mean(), inplace=True)
```

In [86]:
```
land.fatalities.isnull().sum()
```

Out[86]:
```
0
```

**Which is the most landslide prone month.**

In [88]:
```
land.head()
```

Out[88]:

| | id | date | time | country_name | state/province | population | landslide_type | trigger | fa |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 34 | 3/2/2007 | Night | United States | Virginia | 16000 | Landslide | Rain | 1. |
| **1** | 42 | 3/22/2007 | Not Known | United States | Ohio | 17288 | Landslide | Rain | 1. |
| **2** | 56 | 4/6/2007 | Not Known | United States | Pennsylvania | 15930 | Landslide | Rain | 1. |
| **3** | 59 | 4/14/2007 | Not Known | Canada | Quebec | 42786 | Riverbank collapse | Rain | 1. |
| **4** | 61 | 4/15/2007 | Not Known | United States | Kentucky | 6903 | Landslide | Downpour | 0. |

In [92]:
```
land.date = pd.to_datetime(land.date, format="%y%m%d")
land.info()
```

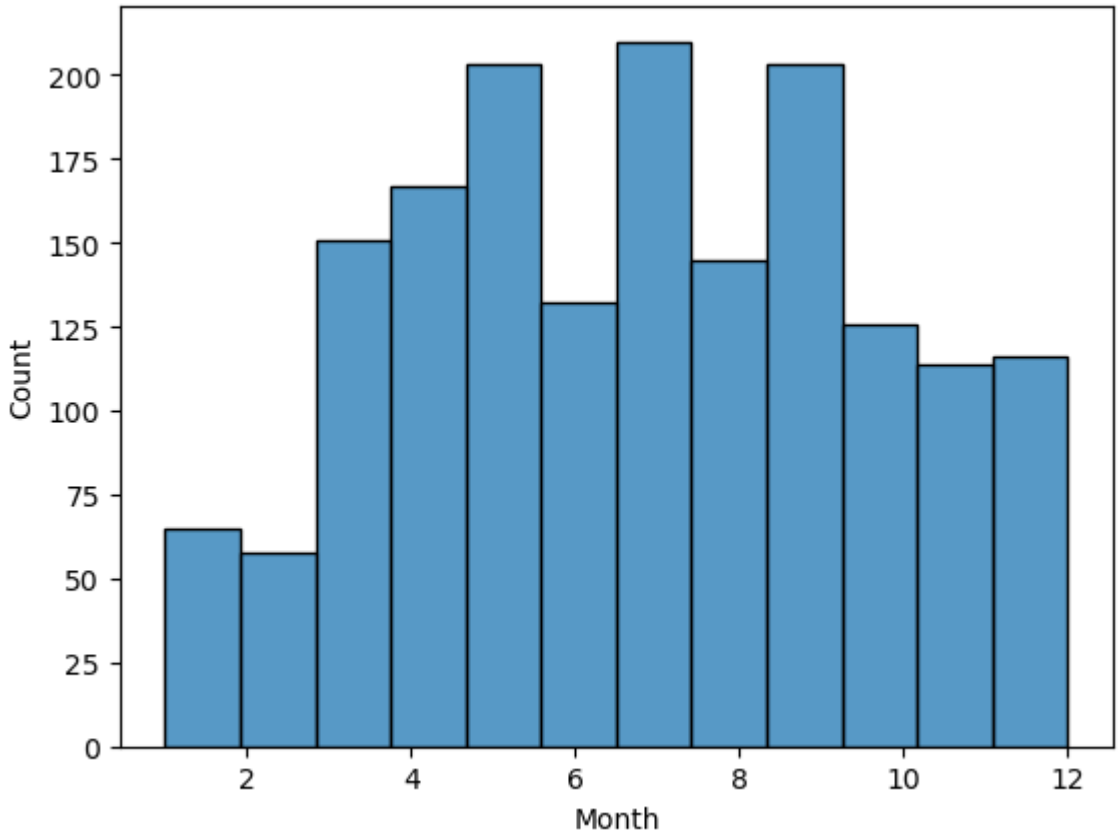
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1690 entries, 0 to 1692
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              1690 non-null   int64
 1   date            1690 non-null   datetime64[ns]
 2   time            1690 non-null   object
 3   country_name    1690 non-null   object
 4   state/province  1689 non-null   object
 5   population      1690 non-null   int64
 6   landslide_type  1689 non-null   object
 7   trigger         1688 non-null   object
 8   fatalities      1690 non-null   float64
 9   Month           1690 non-null   int64
dtypes: datetime64[ns](1), float64(1), int64(3), object(5)
memory usage: 145.2+ KB
```

In [93]:
```
land["Month"] = land.date.dt.month
land.head()
```

Out[93]:

| | id | date | time | country_name | state/province | population | landslide_type | trigger | fatalit |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 34 | 2007-03-02 | Night | United States | Virginia | 16000 | Landslide | Rain | 1.462 |
| **1** | 42 | 2007-03-22 | Not Known | United States | Ohio | 17288 | Landslide | Rain | 1.462 |
| **2** | 56 | 2007-04-06 | Not Known | United States | Pennsylvania | 15930 | Landslide | Rain | 1.462 |
| **3** | 59 | 2007-04-14 | Not Known | Canada | Quebec | 42786 | Riverbank collapse | Rain | 1.462 |
| **4** | 61 | 2007-04-15 | Not Known | United States | Kentucky | 6903 | Landslide | Downpour | 0.000 |

In [97]:
```
sns.histplot(land.Month, bins=12)
plt.show()
```




In [98]:
```
land.Month.value_counts()
```

Out[98]:
```
7     210
5     203
9     203
4     167
3     151
8     145
6     132
10    126
12    116
11    114
1      65
2      58
Name: Month, dtype: int64
```

***We can see July has highes number of landslide***

In [101]:
```
land.time.value_counts()
```

Out[101]:
```
Not Known        1065
Night              97
Morning            87
Afternoon          58
Early morning      36
                 ...
3:20:00             1
1:13                1
9:40:00             1
11:50:00            1
21:06               1
Name: time, Length: 159, dtype: int64
```

In [129]:
```
# Formating the time column by removing ":"

def format_time(x):
    if ":" in x.lower():
        if int(x.split(":")[0]) >= 12 and int(x.split(":")[0]) < 18:
            x = "Afternoon"
        elif int(x.split(":")[0]) < 12:
            x = "Morning"
        elif int(x.split(":")[0]) >= 18:
            x = "Night"
    elif "morning" in x.lower() or "dawn" in x.lower():
        x = "Morning"
    elif "afternoon" in x.lower():
        x = "Afternoon"
    elif "night" in x.lower():
        x = "Night"
    else:
        x = "Not Known"
    return x
```

In [130]:
```
land.time = land.time.apply(format_time)
```

In [131]:
```
land.time.value_counts()
```

Out[131]:
```
Not Known    1086
Morning       265
Night         194
Afternoon     145
Name: time, dtype: int64
```

**Extract customer who are married from the customer's data**

In [132]:
```
marketing.head()
```

Out[132]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer |
|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | $84,835.00 | 0 | 0 | 6/16/14 |
| **1** | 1 | 1961 | Graduation | Single | $57,091.00 | 0 | 0 | 6/15/14 |
| **2** | 10476 | 1958 | Graduation | Married | $67,267.00 | 0 | 1 | 5/13/14 |
| **3** | 1386 | 1967 | Graduation | Together | $32,474.00 | 1 | 1 | 5/11/14 |
| **4** | 5371 | 1989 | Graduation | Single | $21,474.00 | 1 | 0 | 4/8/14 |

In [133]:
```
marketing["Marital_Status"].value_counts()
```

Out[133]:
```
Married     864
Together    580
Single      480
Divorced    232
Widow        77
Alone         3
YOLO          2
Absurd        2
Name: Marital_Status, dtype: int64
```

In [135]:
```
marketing[marketing["Marital_Status"] == "Married"]
```

Out[135]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Custom |
|---|---|---|---|---|---|---|---|---|
| **2** | 10476 | 1958 | Graduation | Married | $67,267.00 | 0 | 1 | 5/13/ |
| **6** | 4073 | 1954 | 2n Cycle | Married | $63,564.00 | 0 | 0 | 1/29/ |
| **8** | 4047 | 1954 | PhD | Married | $65,324.00 | 0 | 1 | 1/11/ |
| **9** | 9477 | 1954 | PhD | Married | $65,324.00 | 0 | 1 | 1/11/ |
| **10** | 2079 | 1947 | 2n Cycle | Married | $81,044.00 | 0 | 0 | 12/27/ |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **2229** | 2106 | 1974 | 2n Cycle | Married | $20,130.00 | 0 | 0 | 3/17/ |
| **2230** | 3363 | 1974 | 2n Cycle | Married | $20,130.00 | 0 | 0 | 3/17/ |
| **2236** | 5263 | 1977 | 2n Cycle | Married | $31,056.00 | 1 | 0 | 1/22/ |
| **2238** | 528 | 1978 | Graduation | Married | $65,819.00 | 0 | 0 | 11/29/ |
| **2239** | 4070 | 1969 | PhD | Married | $94,871.00 | 0 | 2 | 9/1/ |

864 rows × 28 columns

**Extract a customers without a partner and who are born after 1990**

In [157]:
```
temp = marketing[~marketing["Marital_Status"].isin(["Married","Together"])]
temp[temp["Year_Birth"] > 1990]
```

Out[157]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Cus |
|---|---|---|---|---|---|---|---|---|
| **173** | 3005 | 1992 | Graduation | Single | $83,528.00 | 0 | 0 | |
| **293** | 5080 | 1993 | Graduation | Single | $70,515.00 | 0 | 0 | 10 |
| **318** | 7600 | 1992 | Basic | Single | $15,253.00 | 1 | 0 | 10 |
| **639** | 10343 | 1991 | 2n Cycle | Single | $61,618.00 | 0 | 0 | 9 |
| **671** | 5735 | 1991 | Master | Single | $90,638.00 | 0 | 0 | 2 |
| **672** | 5350 | 1991 | Master | Single | $90,638.00 | 0 | 0 | 2 |
| **687** | 10619 | 1994 | Graduation | Single | $95,529.00 | 0 | 0 | 1 |
| **697** | 10548 | 1995 | Graduation | Single | $71,163.00 | 0 | 0 | |
| **744** | 569 | 1991 | Graduation | Single | $90,273.00 | 0 | 0 | 12 |
| **924** | 7431 | 1991 | PhD | Single | $68,126.00 | 0 | 0 | 11 |
| **932** | 4055 | 1992 | Basic | Single | $18,746.00 | 1 | 0 | 5 |

**Extract customers who have more than 5 web purchases.**

In [161]:
```
marketing[marketing.NumWebPurchases > 5]
```

Out[161]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Custom |
|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1961 | Graduation | Single | $57,091.00 | 0 | 0 | 6/15/ |
| **6** | 4073 | 1954 | 2n Cycle | Married | $63,564.00 | 0 | 0 | 1/29/ |
| **8** | 4047 | 1954 | PhD | Married | $65,324.00 | 0 | 1 | 1/11/ |
| **9** | 9477 | 1954 | PhD | Married | $65,324.00 | 0 | 1 | 1/11/ |
| **14** | 10311 | 1969 | Graduation | Married | $4,428.00 | 0 | 1 | 10/5/ |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **2221** | 3846 | 1974 | Graduation | Married | $42,557.00 | 0 | 1 | 8/29/ |
| **2223** | 2831 | 1976 | Graduation | Together | $78,416.00 | 0 | 1 | 6/27/ |
| **2234** | 9977 | 1973 | Graduation | Divorced | $78,901.00 | 0 | 1 | 9/17/ |
| **2237** | 22 | 1976 | Graduation | Divorced | $46,310.00 | 1 | 0 | 12/3/ |
| **2239** | 4070 | 1969 | PhD | Married | $94,871.00 | 0 | 2 | 9/1/ |

628 rows × 28 columns

In [227]:
```
temp = marketing[marketing.Education.isin(["Graduation","PhD","Master"])]
temp[(temp["Income"] > 65000) & (temp["Kidhome"] < 1)]
```

Out[227]:

| | ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer |
|---|---|---|---|---|---|---|---|---|
| **0** | 1826 | 1970 | Graduation | Divorced | 84835.0 | 0 | 0 | 6/16/2014 |
| **2** | 10476 | 1958 | Graduation | Married | 67267.0 | 0 | 1 | 5/13/2014 |
| **5** | 7348 | 1958 | PhD | Single | 71691.0 | 0 | 0 | 3/17/2014 |
| **8** | 4047 | 1954 | PhD | Married | 65324.0 | 0 | 1 | 1/11/2014 |
| **9** | 9477 | 1954 | PhD | Married | 65324.0 | 0 | 1 | 1/11/2014 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **2226** | 1743 | 1974 | Graduation | Single | 69719.0 | 0 | 0 | 5/26/2014 |
| **2234** | 9977 | 1973 | Graduation | Divorced | 78901.0 | 0 | 1 | 9/17/2013 |
| **2235** | 10142 | 1976 | PhD | Divorced | 66476.0 | 0 | 1 | 3/7/2013 |
| **2238** | 528 | 1978 | Graduation | Married | 65819.0 | 0 | 0 | 11/29/2012 |
| **2239** | 4070 | 1969 | PhD | Married | 94871.0 | 0 | 2 | 9/1/2012 |

575 rows × 28 columns

**Find the average amount of product purchased based on each martial status group**

In [228]:
```
marketing.groupby("Marital_Status")[["MntWines","MntFruits","MntMeatProducts",
```

Out[228]:

| Marital_Status | MntWines | MntFruits | MntMeatProducts | MntFishProducts | MntSweetProducts | M |
|---|---|---|---|---|---|---|
| **Absurd** | 355.500000 | 84.500000 | 312.500000 | 205.500000 | 30.500000 | |
| **Alone** | 184.666667 | 4.000000 | 26.333333 | 7.666667 | 7.000000 | |
| **Divorced** | 324.844828 | 27.426724 | 150.206897 | 35.043103 | 26.818966 | |
| **Married** | 299.480324 | 25.734954 | 160.681713 | 35.380787 | 26.701389 | |
| **Single** | 288.331250 | 26.835417 | 182.108333 | 38.216667 | 27.262500 | |
| **Together** | 306.825862 | 25.350000 | 168.103448 | 38.991379 | 26.122414 | |
| **Widow** | 369.272727 | 33.090909 | 189.285714 | 51.389610 | 39.012987 | |
| **YOLO** | 322.000000 | 3.000000 | 50.000000 | 4.000000 | 3.000000 | |

**Median Income of Customer by education and Marital Status**

In [232]:
```
marketing.groupby(["Education","Marital_Status"])["Income"].median()
```

Out[232]:
```
Education   Marital_Status
2n Cycle    Divorced          49118.0
            Married           46462.5
            Single            48668.5
            Together          45774.0
            Widow             47682.0
Basic       Divorced           9548.0
            Married           22352.0
            Single            16383.0
            Together          23179.0
            Widow             22123.0
Graduation  Absurd            79244.0
            Alone             34176.0
            Divorced          55635.0
            Married           50737.0
            Single            49973.5
            Together          53977.0
            Widow             58275.0
Master      Absurd            65487.0
```

In [253]:
```
stock = pd.read_csv("Stock.csv")
stock.head()
```

Out[253]:

| | Unnamed: 0 | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|---|
| **0** | 1/2/2003 0:00 | 7.40 | 21.11 | 29.22 | 909.03 |
| **1** | 1/3/2003 0:00 | 7.45 | 21.14 | 29.24 | 908.59 |
| **2** | 1/6/2003 0:00 | 7.45 | 21.52 | 29.96 | 929.01 |
| **3** | 1/7/2003 0:00 | 7.43 | 21.93 | 28.95 | 922.93 |
| **4** | 1/8/2003 0:00 | 7.28 | 21.31 | 28.83 | 909.93 |

**Calculate the Yearly stock return for four organizations**

In [254]:
```
stock.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2214 entries, 0 to 2213
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  2214 non-null   object
 1   AAPL        2214 non-null   float64
 2   MSFT        2214 non-null   float64
 3   XOM         2214 non-null   float64
 4   SPX         2214 non-null   float64
dtypes: float64(4), object(1)
memory usage: 86.6+ KB
```

In [260]:
```
stock.rename(columns = {'Unnamed: 0':'Date'}, inplace = True)
stock.Date = pd.to_datetime(stock.Date) # Conveting into date date type
```

In [263]:
```
stock["Year"] = stock.Date.dt.year
stock.head()
```

Out[263]:

| | Date | AAPL | MSFT | XOM | SPX | Year |
|---|---|---|---|---|---|---|
| **0** | 2003-01-02 | 7.40 | 21.11 | 29.22 | 909.03 | 2003 |
| **1** | 2003-01-03 | 7.45 | 21.14 | 29.24 | 908.59 | 2003 |
| **2** | 2003-01-06 | 7.45 | 21.52 | 29.96 | 929.01 | 2003 |
| **3** | 2003-01-07 | 7.43 | 21.93 | 28.95 | 922.93 | 2003 |
| **4** | 2003-01-08 | 7.28 | 21.31 | 28.83 | 909.93 | 2003 |

In [268]:
```
stock.groupby("Year").mean()
```

Out[268]:

| Year | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2003** | 9.272619 | 20.595119 | 30.211111 | 965.227540 |
| **2004** | 17.763889 | 21.850437 | 38.875437 | 1130.649444 |
| **2005** | 46.675952 | 23.072421 | 51.045476 | 1207.229444 |
| **2006** | 70.810637 | 23.759363 | 58.458406 | 1310.461633 |
| **2007** | 128.273904 | 27.904422 | 75.767131 | 1477.184343 |
| **2008** | 141.979012 | 24.760593 | 76.525968 | 1220.042055 |
| **2009** | 146.814127 | 21.885397 | 67.124960 | 948.046389 |
| **2010** | 259.842460 | 26.262619 | 63.067976 | 1139.965516 |
| **2011** | 356.526834 | 25.825930 | 79.042663 | 1276.093015 |