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

```
df = pd.read_csv('Ecommerce Purchases')
```

```
df
```

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code	Pr
0	16629 Pace Camp Apt. 448\nAlexisborough, NE 77...	46 in	PM	Opera/9.56.(X11; Linux x86_64; sl-SI) Presto/2...	Martinez-Herman	6011929061123406	02/20	900	
1	9374 Jasmine Spurs Suite 508\nSouth John, TN 8...	28 rn	PM	Opera/8.93.(Windows 98; Win 9x 4.90; en-US) Pr...	Fletcher, Richards and Whitaker	3337758169645356	11/18	561	Ma
2	Unit 0065 Box 5052\nDPO AP 27450	94 vE	PM	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Simpson, Williams and Pham	675957666125	08/19	699	
3	7780 Julia Fords\nNew Stacy, WA 45798	36 vm	PM	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_0 ...	Williams, Marshall and Buchanan	6011578504430710	02/24	384	[
4	23012 Munoz Drive Suite 337\nNew Cynthia, TX 5...	20 IE	AM	Opera/9.58.(X11; Linux x86_64; it-IT) Presto/2...	Brown, Watson and Andrews	6011456623207998	10/25	678	
...	...	...	...	...	...	...	...	...	
9995	966 Castaneda Locks\nWest Juliafurt, CO 96415	92 XI	PM	Mozilla/5.0 (Windows NT 5.1) AppleWebKit/5352 ...	Randall-Sloan	342945015358701	03/22	838	
9996	832 Curtis Dam Suite 785\nNorth Edwardburgh, T...	41 JY	AM	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Hale, Collins and Wilson	210033169205009	07/25	207	
9997	Unit 4434 Box 6343\nDPO AE 28026-0283	74 Zh	AM	Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_7...	Anderson Ltd	6011539787356311	05/21	1	
9998	0096 English Rest\nRoystad, IA 12457	74 cL	PM	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_8;...	Cook Inc	180003348082930	11/17	987	A
9999	40674 Barrett Stravenue\nGrimesville, WI 79682	64 Hr	AM	Mozilla/5.0 (X11; Linux i686; rv:1.9.5.20) Gec...	Greene Inc	4139972901927273	02/19	302	

10000 rows × 14 columns

```
#display the top ten rows of the dataset

df.head(10)
```

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code
0	16629 Pace Camp Apt. 448\nAlexisborough, NE 77...	46 in	PM	Opera/9.56.(X11; Linux x86_64; sl-SI) Presto/2...	Martinez-Herman	6011929061123406	02/20	900
1	9374 Jasmine Spurs Suite 508\nSouth John, TN 8...	28 rn	PM	Opera/8.93.(Windows 98; Win 9x 4.90; en-US) Pr...	Fletcher, Richards and Whitaker	3337758169645356	11/18	561
2	Unit 0065 Box 5052\nDPO AP 27450	94 vE	PM	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Simpson, Williams and Pham	675957666125	08/19	699
3	7780 Julia Fords\nNew Stacy, WA 45798	36 vm	PM	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_0 ...	Williams, Marshall and Buchanan	6011578504430710	02/24	384
4	23012 Munoz Drive Suite 337\nNew Cynthia, TX 5...	20 IE	AM	Opera/9.58.(X11; Linux x86_64; it-IT) Presto/2...	Brown, Watson and Andrews	6011456623207998	10/25	678
5	7502 Powell Mission Apt. 768\nTravisland, VA 3...	21 XT	PM	Mozilla/5.0 (Macintosh; U; PPC Mac OS X 10_8_5...	Silva-Anderson	30246185196287	07/25	7169
6	93971 Conway Causeway\nAndersonburgh, AZ 75107	96 Xt	AM	Mozilla/5.0 (compatible; MSIE 7.0; Windows NT ...	Gibson and Sons	6011398782655569	07/24	714
7	260 Rachel Plains Suite 366\nCastroberg, WV 24...	96 pG	PM	Mozilla/5.0 (X11; Linux i686) AppleWebKit/5350...	Marshall-Collins	561252141909	06/25	256
8	2129 Dylan Burg\nNew	45	PM	Mozilla/5.0 (Macintosh; U; Intel	Galloway	180041795790001	04/24	899

```
#display the last ten rows of the dataset

df.tail(10)
```

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code	Pro
9990	75731 Molly Springs\nWest Danielle, VT 96934-5102	93 ty	PM	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4;...	Pace, Vazquez and Richards	869968197049750	04/24	877	ˇ
9991	PSC 8165, Box 8498\nAPO AP 60327-0346	50 dA	AM	Mozilla/5.0 (compatible; MSIE 8.0; Windows NT ...	Snyder Inc	4221582137197481	02/24	969	Vo
				Mozilla/5.0					

```
#check the datatype of each coloumn

df.dtypes
```

```
Address              object
Lot                  object
AM or PM             object
Browser Info         object
Company              object
Credit Card           int64
CC Exp Date          object
CC Security Code      int64
CC Provider          object
Email                object
Job                  object
IP Address           object
Language             object
Purchase Price      float64
dtype: object
```

Edwardburgh, T... ... Wilson

```
#checking the number of null values in the dataset

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

```
Address             0
Lot                 0
AM or PM            0
Browser Info        0
Company             0
Credit Card         0
CC Exp Date         0
CC Security Code    0
CC Provider         0
Email               0
Job                 0
IP Address          0
Language            0
Purchase Price      0
dtype: int64
```

```
#check null values in a particular column in the dataset

df['Purchase Price'].isnull().sum()
```

```
0
```

```
#check the number of rows and columns in the dataset
df.shape
```

```
(10000, 14)
```

```
#just check the number of rows in the dataset
len(df)
```

```
10000
```

```
#just check the number of columns in the dataset
len(df.columns)
```

```
14
```

```
#display the information of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Address           10000 non-null  object 
 1   Lot               10000 non-null  object 
 2   AM or PM          10000 non-null  object 
 3   Browser Info      10000 non-null  object 
 4   Company           10000 non-null  object 
 5   Credit Card       10000 non-null  int64  
 6   CC Exp Date       10000 non-null  object 
 7   CC Security Code  10000 non-null  int64  
 8   CC Provider       10000 non-null  object 
 9   Email             10000 non-null  object 
 10  Job               10000 non-null  object 
 11  IP Address        10000 non-null  object 
 12  Language          10000 non-null  object 
 13  Purchase Price    10000 non-null  float64
dtypes: float64(1), int64(2), object(11)
memory usage: 1.1+ MB
```

```
#retreive the highest purchase value from the dataset
sorted_descending = df.sort_values('Purchase Price', ascending = False).reset_index()
```

```
sorted_descending.loc[0]
```

```
index                                                            2092
Address                      63773 Shelton Greens\nAshleyton, MA 00493
Lot                                                             56 lu
AM or PM                                                           AM
Browser Info        Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/53...
Company                                                   Pitts Group
Credit Card                                             4292741269160
CC Exp Date                                                     06/18
CC Security Code                                                  824
CC Provider                                                   Maestro
Email                                        heatherwoodard@lloyd.com
Job                                            Surveyor, hydrographic
IP Address                                             172.197.216.229
Language                                                           el
Purchase Price                                                  99.99
Name: 0, dtype: object
```

```
#another way to get the highest value of a particular column from the dataset

df['Purchase Price'].max()
```

```
99.99
```

```
#get the lowest value of a particular column in the dataset

sorted_ascending = df.sort_values('Purchase Price', ascending = True).reset_index()
```

```
sorted_ascending.head(5)
```

	index	Address	Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code	Provid
0	5487	465 Mallory Ways\nNorth Rebecca, RI 82734-1160	93 OH	PM	Mozilla/5.0 (X11; Linux i686; rv:1.9.6.20) Gec...	Flynn and Sons	30469912089738	09/23	236	Discov
1	2876	332 Jones Parkways\nEast Katherineville, GA 64230	39 GT	AM	Mozilla/5.0 (Macintosh; U; PPC Mac OS X	Lyons, Diaz and Clark	4204500444841766	01/18	38	VISA d

```
sorted_ascending.loc[0]
```

```
index                                                        5487
Address                465 Mallory Ways\nNorth Rebecca, RI 82734-1160
Lot                                                         93 OH
AM or PM                                                       PM
Browser Info        Mozilla/5.0 (X11; Linux i686; rv:1.9.6.20) Gec...
Company                                            Flynn and Sons
Credit Card                                        30469912089738
CC Exp Date                                                 09/23
CC Security Code                                              236
CC Provider                                              Discover
Email                                         mjohnson@austin.org
Job                                                 Stage manager
IP Address                                            43.99.56.59
Language                                                       zh
Purchase Price                                                0.0
Name: 0, dtype: object
```

```
 #another way to get the lowest value from a particular column in the dataset

df['Purchase Price'].min()
```

```
0.0
```

```
#get the mean value of a particular column of a given dataset

df['Purchase Price'].mean()
```

```
50.347302
```

```
#checking for matching values for a particular column within a dataset
#in this case, we are finding the number of people of have French as their language

df.columns
```

```
Index(['Address', 'Lot', 'AM or PM', 'Browser Info', 'Company', 'Credit Card',
       'CC Exp Date', 'CC Security Code', 'CC Provider', 'Email', 'Job',
       'IP Address', 'Language', 'Purchase Price'],
      dtype='object')
```

```
#check if there are any missing values within that column
df['Language'].isnull().sum()
```

```
0
```

```
#check what are the different unique values present in the dataset
df['Language'].unique()
```

```
array(['el', 'fr', 'de', 'es', 'ru', 'pt', 'zh', 'en', 'it'], dtype=object)
```

```
#check the number of unique values present in the dataset
df['Language'].nunique()
```

```
9
```

```
#group the number of unique values with their respective frequencies
df.groupby('Language').nunique()
```

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code	CC Provider	Email	Job	IP Address
Language												
de	1155	1151	2	1086	1113	1155	121	736	10	1155	525	1155
el	1137	1134	2	1074	1104	1137	121	739	10	1137	511	1137
en	1098	1096	2	1041	1071	1098	121	732	10	1098	518	1098
es	1095	1094	2	1047	1071	1095	121	717	10	1095	516	1095
fr	1097	1096	2	1042	1069	1097	121	720	10	1097	515	1097
it	1086	1083	2	1032	1059	1086	121	711	10	1086	513	1086
pt	1118	1116	2	1046	1085	1118	121	740	10	1118	521	1118

```
len(df[df['Language'] == 'fr'])
```

```
1097
```

```
df[df['Language'] == 'fr'].count()
```

```
Address             1097
Lot                 1097
AM or PM            1097
Browser Info        1097
Company             1097
Credit Card         1097
CC Exp Date         1097
CC Security Code    1097
CC Provider         1097
Email               1097
Job                 1097
IP Address          1097
Language            1097
Purchase Price      1097
dtype: int64
```

```
#get the job title than contains engineer
```

```
df.columns
```

```
Index(['Address', 'Lot', 'AM or PM', 'Browser Info', 'Company', 'Credit Card',
       'CC Exp Date', 'CC Security Code', 'CC Provider', 'Email', 'Job',
       'IP Address', 'Language', 'Purchase Price'],
      dtype='object')
```

```
len(df[df['Job'].str.contains('engineer', case = True)])
```

```
531
```

```
#find email of the person with the following IP address - 132.207.160.22
```

```
df.columns
```

```
Index(['Address', 'Lot', 'AM or PM', 'Browser Info', 'Company', 'Credit Card',
       'CC Exp Date', 'CC Security Code', 'CC Provider', 'Email', 'Job',
       'IP Address', 'Language', 'Purchase Price'],
      dtype='object')
```

```
df[df['IP Address'] == '132.207.160.22']
```

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code	CC Provider	Emai
2	Unit 0065 Box 5052\nDPO AP 27450	94 vE	PM	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Simpson, Williams and Pham	675957666125	08/19	699	JCB 16 digit	amymiller@morales harrison.cor

```
#how many people have a mastercard as their credit card provider and made a purchase above 50

len(df[(df['CC Provider'] == 'Mastercard') & (df['Purchase Price'] > 50)])
```

```
405
```

```
#find the email of the person with the following credit card number - 675957666125
df[df['Email'] == "amymiller@morales-harrison.com"]['Job']
```

```
2    Customer service manager
Name: Job, dtype: object
```

```
# how many people purchase during the AM and how many people purchase during the PM

df.columns
```

```
Index(['Address', 'Lot', 'AM or PM', 'Browser Info', 'Company', 'Credit Card',
       'CC Exp Date', 'CC Security Code', 'CC Provider', 'Email', 'Job',
       'IP Address', 'Language', 'Purchase Price'],
      dtype='object')
```

```
print('Number of AM people:', len(df[df['AM or PM'] == 'AM']))
print('Number of PM people:', len(df[df['AM or PM'] == 'PM']))
```

```
Number of AM people: 4932
Number of PM people: 5068
```

```
#how many people have a credit card that expires in 2020

df.columns
```

```
Index(['Address', 'Lot', 'AM or PM', 'Browser Info', 'Company', 'Credit Card',
       'CC Exp Date', 'CC Security Code', 'CC Provider', 'Email', 'Job',
       'IP Address', 'Language', 'Purchase Price'],
      dtype='object')
```

```
def fun():
  count = 0
  for date in df['CC Exp Date']:
    if date.split('/')[1] == '20':
      count = count + 1
  print(count)

fun()
```

```
988
```

```
len(df[df['CC Exp Date'].apply(lambda x: x[3:] == '20')])
```

```
988
```

```
#top five most popular email providers

list1 = []
for email in df['Email']:
  list1.append(email.split('@')[1])
```

```
new_frame = df['Temp'] = list1
```

```
df.head(1)
```

```
df['Temp'].value_counts().head(5)

     hotmail.com     1638
     yahoo.com       1616
     gmail.com       1605
     smith.com         42
     williams.com      37
     Name: Temp, dtype: int64
```

```
df['Email'].apply(lambda x:x.split('@')[1]).value_counts().head(5)

     hotmail.com     1638
     yahoo.com       1616
     gmail.com       1605
     smith.com         42
     williams.com      37
     Name: Email, dtype: int64
```

✓ 0s completed at 4:58 PM