Ecommerce Purchases Project:-

This is a simple project in which the Exploratory Data Analysis of the given dataset is performed using Pandas library.

Link of the dataset: https://www.kaggle.com/datasets/utkarsharya/ecommerce-purchases

```
import pandas as pd
data= pd.read_csv('/content/Ecommerce Purchases.zip')
df=data.copy()
```

```
#Printing the first 10 rows of the data set
df.head(10)
```

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	
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	0
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	1
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	0
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	0
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	1
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	0
6	93971 Conway Causeway\nAndersonburgh, AZ 75107	96 Xt	AM	Mozilla/5.0 (compatible; MSIE 7.0; Windows NT ...	Gibson and Sons	6011398782655569	0
7	260 Rachel Plains Suite 366\nCastroberg, WV 24...	96 pG	PM	Mozilla/5.0 (X11; Linux i686) AppleWebKit/5350...	Marshall-Collins	561252141909	0
8	2129 Dylan Burg\nNew Michelle, ME 28650	45 JN	PM	Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_7...	Galloway and Sons	180041795790001	0
9	3795 Dawson Extensions\nLake Tinafort, ID 88739	15 Ug	AM	Mozilla/5.0 (X11; Linux i686; rv:1.9.7.20) Gec...	Rivera, Buchanan and Ramirez	4396283918371	0

```
#displaying the last 10 rows of the data set
df.tail(10)
```

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code	Provi
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	JCI
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	Voy
9992	885 Allen Mountains Apt. 230\nWallhaven, LA 16995	40 vH	PM	Mozilla/5.0 (Macintosh; PPC Mac OS X 10_6_5) A...	Wells Ltd	4664825258997302	10/20	431	Disc
9993	7555 Larson Locks Suite 229\nEllisburgh, MA 34...	72 jg	PM	Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_8...	Colon and Sons	30025560104631	10/25	629	Mae
9994	6276 Rojas Hollow\nLake Louis, WY 56410-7837	93 Ex	PM	Opera/9.68.(X11; Linux x86_64; sl-SI) Presto/2...	Ritter-Smith	3112186784121077	01/25	1823	Mae
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	JCI
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	JCI

```
#Checking the data type of each column
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
```

```
#Checking null values in the data set
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
```

```
#Counting the number of rows and columns in the data set
print("No. of rows in the data set is: ", len(df))
print("No of columns in the data set is: ", len(df.columns))
df.info()
```

```
No. of rows in the data set is:  10000
No of columns in the data set is:  14
<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
```

```
#Finding the highest and lowest purchase price
print("Highest Purchase Price: ", df['Purchase Price'].max())
print("Lowest Purchase Price: ", df['Purchase Price'].min())
```

```
Highest Purchase Price:  99.99
Lowest Purchase Price:  0.0
```

```
#Printing the average purchase price
print("Highest Purchase Price: ", df['Purchase Price'].mean())
```

```
Highest Purchase Price:  50.347302
```

```
#Finding the number of people having french as their language
print("Number of people having French as their language is: ",len(df[df['Language']=='fr']))
```

```
Number of people having French as their language is:  1097
```

```
#Finding the job title that contains Engineer
print("The no. of people who are engineers are:",len(df[df['Job'].str.contains('engineer', case= False)]))
```

```
The no. of people who are engineers are: 984
```

```
#Finding the email of the person with the following IP address : 132.207.160.22
df[df['IP Address']== '132.207.160.22']['Email']
```

```
2    amymiller@morales-harrison.com
Name: Email, dtype: object
```

```
#How many people have Mastercard as their Credit card provider and made a purchase above 50?
print(len(df[(df['CC Provider']== 'Mastercard')  & (df["Purchase Price"]> 50)]))
```

```
405
```

```
#Finding the email of the person with the following credit card no: 4664825258997302
df[df['Credit Card'] ==4664825258997302 ] ['Email']
```

```
9992    bberry@wright.net
Name: Email, dtype: object
```

```
#How many people purchase during the AM and how many during the PM?
print("No of people who have purchased during AM: ",len(df[df["AM or PM"]=='AM']))
print("No of people who have purchased during PM: ",len(df[df["AM or PM"]=='PM']))
```

```
No of people who have purchased during AM:  4932
No of people who have purchased during PM:  5068
```

```
#How many people have a credit card that expires in 2020?
len(df[df['CC Exp Date'].str.contains("/20", case=False)])
```

```
988
```

```
#Printing the top 5 popular email providers
l=[]
for email in df['Email']:
  l.append(email.split('@')[1])

df['temp']=l
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
```

```
#The other way to simply get the above is:
df["Email"].apply(lambda x:x.split('@')[1]).value_counts().head()
```

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