Tasks

**import** pandas **as** pd

df**=**pd**.**read\_csv("C:\\Users\\y20cs127\\Desktop\\supermarket\_sales.csv")

print(df)

Invoice ID Branch City Customer type Gender \

0 750-67-8428 A Yangon Member Female

1 226-31-3081 C Naypyitaw Normal Female

2 631-41-3108 A Yangon Normal Male

3 123-19-1176 A Yangon Member Male

4 373-73-7910 A Yangon Normal Male

.. ... ... ... ... ...

995 233-67-5758 C Naypyitaw Normal Male

996 303-96-2227 B Mandalay Normal Female

997 727-02-1313 A Yangon Member Male

998 347-56-2442 A Yangon Normal Male

999 849-09-3807 A Yangon Member Female

Product line Unit price Quantity Tax 5% Total \

0 Health and beauty 74.69 7 26.1415 548.9715

1 Electronic accessories 15.28 5 3.8200 80.2200

2 Home and lifestyle 46.33 7 16.2155 340.5255

3 Health and beauty 58.22 8 23.2880 489.0480

4 Sports and travel 86.31 7 30.2085 634.3785

.. ... ... ... ... ...

995 Health and beauty 40.35 1 2.0175 42.3675

996 Home and lifestyle 97.38 10 48.6900 1022.4900

997 Food and beverages 31.84 1 1.5920 33.4320

998 Home and lifestyle 65.82 1 3.2910 69.1110

999 Fashion accessories 88.34 7 30.9190 649.2990

Date Time Payment cogs gross margin percentage \

0 1/5/2019 13:08 Ewallet 522.83 4.761905

1 3/8/2019 10:29 Cash 76.40 4.761905

2 3/3/2019 13:23 Credit card 324.31 4.761905

3 1/27/2019 20:33 Ewallet 465.76 4.761905

4 2/8/2019 10:37 Ewallet 604.17 4.761905

.. ... ... ... ... ...

995 1/29/2019 13:46 Ewallet 40.35 4.761905

996 3/2/2019 17:16 Ewallet 973.80 4.761905

997 2/9/2019 13:22 Cash 31.84 4.761905

998 2/22/2019 15:33 Cash 65.82 4.761905

999 2/18/2019 13:28 Cash 618.38 4.761905

gross income Rating

0 26.1415 9.1

1 3.8200 9.6

2 16.2155 7.4

3 23.2880 8.4

4 30.2085 5.3

.. ... ...

995 2.0175 6.2

996 48.6900 4.4

997 1.5920 7.7

998 3.2910 4.1

999 30.9190 6.6

[1000 rows x 17 columns]

1. Output the first few rows of data

df[0:7]

|  | **Invoice ID** | **Branch** | **City** | **Customer type** | **Gender** | **Product line** | **Unit price** | **Quantity** | **Tax 5%** | **Total** | **Date** | **Time** | **Payment** | **cogs** | **gross margin percentage** | **gross income** | **Rating** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 750-67-8428 | A | Yangon | Member | Female | Health and beauty | 74.69 | 7 | 26.1415 | 548.9715 | 1/5/2019 | 13:08 | Ewallet | 522.83 | 4.761905 | 26.1415 | 9.1 |
| **1** | 226-31-3081 | C | Naypyitaw | Normal | Female | Electronic accessories | 15.28 | 5 | 3.8200 | 80.2200 | 3/8/2019 | 10:29 | Cash | 76.40 | 4.761905 | 3.8200 | 9.6 |
| **2** | 631-41-3108 | A | Yangon | Normal | Male | Home and lifestyle | 46.33 | 7 | 16.2155 | 340.5255 | 3/3/2019 | 13:23 | Credit card | 324.31 | 4.761905 | 16.2155 | 7.4 |
| **3** | 123-19-1176 | A | Yangon | Member | Male | Health and beauty | 58.22 | 8 | 23.2880 | 489.0480 | 1/27/2019 | 20:33 | Ewallet | 465.76 | 4.761905 | 23.2880 | 8.4 |
| **4** | 373-73-7910 | A | Yangon | Normal | Male | Sports and travel | 86.31 | 7 | 30.2085 | 634.3785 | 2/8/2019 | 10:37 | Ewallet | 604.17 | 4.761905 | 30.2085 | 5.3 |
| **5** | 699-14-3026 | C | Naypyitaw | Normal | Male | Electronic accessories | 85.39 | 7 | 29.8865 | 627.6165 | 3/25/2019 | 18:30 | Ewallet | 597.73 | 4.761905 | 29.8865 | 4.1 |
| **6** | 355-53-5943 | A | Yangon | Member | Female | Electronic accessories | 68.84 | 6 | 20.6520 | 433.6920 | 2/25/2019 | 14:36 | Ewallet | 413.04 | 4.761905 | 20.6520 | 5.8 |

1. Output data dimensions (i.e. the number of rows and columns of the dataset)

rows**=**len(df**.**axes[0])

cols**=**len(df**.**axes[1])

print("Number of rows are ",rows)

print("Number of columns are ",cols)

*#Another*

print(len(df))

print(len(df**.**columns))

Number of rows are 1000

Number of columns are 17

1000

17

1. Output the column header names

df**.**columns

Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',

'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date',

'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income',

'Rating'],

dtype='object')

1. Obtain detailed information about the dataset

df**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1000 entries, 0 to 999

Data columns (total 17 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Invoice ID 1000 non-null object

1 Branch 1000 non-null object

2 City 1000 non-null object

3 Customer type 1000 non-null object

4 Gender 1000 non-null object

5 Product line 1000 non-null object

6 Unit price 1000 non-null float64

7 Quantity 1000 non-null int64

8 Tax 5% 1000 non-null float64

9 Total 1000 non-null float64

10 Date 1000 non-null object

11 Time 1000 non-null object

12 Payment 1000 non-null object

13 cogs 1000 non-null float64

14 gross margin percentage 1000 non-null float64

15 gross income 1000 non-null float64

16 Rating 1000 non-null float64

dtypes: float64(7), int64(1), object(9)

memory usage: 132.9+ KB

1. Output statistical information about the dataset

df**.**describe()

|  | **Unit price** | **Quantity** | **Tax 5%** | **Total** | **cogs** | **gross margin percentage** | **gross income** | **Rating** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.00000 | 1.000000e+03 | 1000.000000 | 1000.00000 |
| **mean** | 55.672130 | 5.510000 | 15.379369 | 322.966749 | 307.58738 | 4.761905e+00 | 15.379369 | 6.97270 |
| **std** | 26.494628 | 2.923431 | 11.708825 | 245.885335 | 234.17651 | 6.131498e-14 | 11.708825 | 1.71858 |
| **min** | 10.080000 | 1.000000 | 0.508500 | 10.678500 | 10.17000 | 4.761905e+00 | 0.508500 | 4.00000 |
| **25%** | 32.875000 | 3.000000 | 5.924875 | 124.422375 | 118.49750 | 4.761905e+00 | 5.924875 | 5.50000 |
| **50%** | 55.230000 | 5.000000 | 12.088000 | 253.848000 | 241.76000 | 4.761905e+00 | 12.088000 | 7.00000 |
| **75%** | 77.935000 | 8.000000 | 22.445250 | 471.350250 | 448.90500 | 4.761905e+00 | 22.445250 | 8.50000 |
| **max** | 99.960000 | 10.000000 | 49.650000 | 1042.650000 | 993.00000 | 4.761905e+00 | 49.650000 | 10.00000 |

1. Output the third row of the data frame

df**.**iloc[2]

Invoice ID 631-41-3108

Branch A

City Yangon

Customer type Normal

Gender Male

Product line Home and lifestyle

Unit price 46.33

Quantity 7

Tax 5% 16.2155

Total 340.5255

Date 3/3/2019

Time 13:23

Payment Credit card

cogs 324.31

gross margin percentage 4.761905

gross income 16.2155

Rating 7.4

Name: 2, dtype: object

1. Output the last row of the data frame

df**.**iloc[**-**1]

Invoice ID 849-09-3807

Branch A

City Yangon

Customer type Member

Gender Female

Product line Fashion accessories

Unit price 88.34

Quantity 7

Tax 5% 30.919

Total 649.299

Date 2/18/2019

Time 13:28

Payment Cash

cogs 618.38

gross margin percentage 4.761905

gross income 30.919

Rating 6.6

Name: 999, dtype: object

1. Output the last 3 columns of the data frame

df1 **=** df**.**iloc[:,**-**3:]

df1

Out[64]:

|  | **gross margin percentage** | **gross income** | **Rating** |
| --- | --- | --- | --- |
| **0** | 4.761905 | 26.1415 | 9.1 |
| **1** | 4.761905 | 3.8200 | 9.6 |
| **2** | 4.761905 | 16.2155 | 7.4 |
| **3** | 4.761905 | 23.2880 | 8.4 |
| **4** | 4.761905 | 30.2085 | 5.3 |
| **...** | ... | ... | ... |
| **995** | 4.761905 | 2.0175 | 6.2 |
| **996** | 4.761905 | 48.6900 | 4.4 |
| **997** | 4.761905 | 1.5920 | 7.7 |
| **998** | 4.761905 | 3.2910 | 4.1 |
| **999** | 4.761905 | 30.9190 | 6.6 |

1000 rows × 3 columns

1. Output information about the object and bool data types data within the dataset

df**.**select\_dtypes(include**=**['bool','object'])

|  | **Invoice ID** | **Branch** | **City** | **Customer type** | **Gender** | **Product line** | **Date** | **Time** | **Payment** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 750-67-8428 | A | Yangon | Member | Female | Health and beauty | 1/5/2019 | 13:08 | Ewallet |
| **1** | 226-31-3081 | C | Naypyitaw | Normal | Female | Electronic accessories | 3/8/2019 | 10:29 | Cash |
| **2** | 631-41-3108 | A | Yangon | Normal | Male | Home and lifestyle | 3/3/2019 | 13:23 | Credit card |
| **3** | 123-19-1176 | A | Yangon | Member | Male | Health and beauty | 1/27/2019 | 20:33 | Ewallet |
| **4** | 373-73-7910 | A | Yangon | Normal | Male | Sports and travel | 2/8/2019 | 10:37 | Ewallet |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **995** | 233-67-5758 | C | Naypyitaw | Normal | Male | Health and beauty | 1/29/2019 | 13:46 | Ewallet |
| **996** | 303-96-2227 | B | Mandalay | Normal | Female | Home and lifestyle | 3/2/2019 | 17:16 | Ewallet |
| **997** | 727-02-1313 | A | Yangon | Member | Male | Food and beverages | 2/9/2019 | 13:22 | Cash |
| **998** | 347-56-2442 | A | Yangon | Normal | Male | Home and lifestyle | 2/22/2019 | 15:33 | Cash |
| **999** | 849-09-3807 | A | Yangon | Member | Female | Fashion accessories | 2/18/2019 | 13:28 | Cash |

1000 rows × 9 columns

10. The total data type is float, can we get information about the number of interger32?

print(df['Total']**.**head())

df['Total']**=**df['Total']**.**astype('int')

print(df['Total']**.**head())

0 548.9715

1 80.2200

2 340.5255

3 489.0480

4 634.3785

Name: Total, dtype: float64

0 548

1 80

2 340

3 489

4 634

Name: Total, dtype: int32

11. Create a new column in the data frame which ranks the rows in decreasing order

df['Rating\_Rank'] **=** df['Rating']**.**rank(ascending **=** 0)

print(df)

Invoice ID Branch City Customer type Gender \

0 750-67-8428 A Yangon Member Female

1 226-31-3081 C Naypyitaw Normal Female

2 631-41-3108 A Yangon Normal Male

3 123-19-1176 A Yangon Member Male

4 373-73-7910 A Yangon Normal Male

.. ... ... ... ... ...

995 233-67-5758 C Naypyitaw Normal Male

996 303-96-2227 B Mandalay Normal Female

997 727-02-1313 A Yangon Member Male

998 347-56-2442 A Yangon Normal Male

999 849-09-3807 A Yangon Member Female

Product line Unit price Quantity Tax 5% Total \

0 Health and beauty 74.69 7 26.1415 548.9715

1 Electronic accessories 15.28 5 3.8200 80.2200

2 Home and lifestyle 46.33 7 16.2155 340.5255

3 Health and beauty 58.22 8 23.2880 489.0480

4 Sports and travel 86.31 7 30.2085 634.3785

.. ... ... ... ... ...

995 Health and beauty 40.35 1 2.0175 42.3675

996 Home and lifestyle 97.38 10 48.6900 1022.4900

997 Food and beverages 31.84 1 1.5920 33.4320

998 Home and lifestyle 65.82 1 3.2910 69.1110

999 Fashion accessories 88.34 7 30.9190 649.2990

Date Time Payment cogs gross margin percentage \

0 1/5/2019 13:08 Ewallet 522.83 4.761905

1 3/8/2019 10:29 Cash 76.40 4.761905

2 3/3/2019 13:23 Credit card 324.31 4.761905

3 1/27/2019 20:33 Ewallet 465.76 4.761905

4 2/8/2019 10:37 Ewallet 604.17 4.761905

.. ... ... ... ... ...

995 1/29/2019 13:46 Ewallet 40.35 4.761905

996 3/2/2019 17:16 Ewallet 973.80 4.761905

997 2/9/2019 13:22 Cash 31.84 4.761905

998 2/22/2019 15:33 Cash 65.82 4.761905

999 2/18/2019 13:28 Cash 618.38 4.761905

gross income Rating Rating\_Rank

0 26.1415 9.1 144.5

1 3.8200 9.6 63.0

2 16.2155 7.4 421.5

3 23.2880 8.4 261.5

4 30.2085 5.3 785.0

.. ... ... ...

995 2.0175 6.2 635.0

996 48.6900 4.4 924.0

997 1.5920 7.7 371.5

998 3.2910 4.1 981.0

999 30.9190 6.6 560.5

[1000 rows x 18 columns]

12. Output rows which have Nan entries in gross income

selected\_rows **=** df[df['gross income']**.**isnull()]

selected\_rows

|  | **Invoice ID** | **Branch** | **City** | **Customer type** | **Gender** | **Product line** | **Unit price** | **Quantity** | **Tax 5%** | **Total** | **Date** | **Time** | **Payment** | **cogs** | **gross margin percentage** | **gross income** | **Rating** | **Rating\_Rank** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

13.Output rows which do not have Nan entries in order total

----------------------------------------------------

df2**=**pd**.**read\_csv("C:\\Users\\y20cs127\\Desktop\\data.csv")

df2

|  | **Duration** | **Pulse** | **Maxpulse** | **Calories** |
| --- | --- | --- | --- | --- |
| **0** | 60 | 110 | 130 | 409.1 |
| **1** | 60 | 117 | 145 | 479.0 |
| **2** | 60 | 103 | 135 | 340.0 |
| **3** | 45 | 109 | 175 | 282.4 |
| **4** | 45 | 117 | 148 | 406.0 |
| **...** | ... | ... | ... | ... |
| **164** | 60 | 105 | 140 | 290.8 |
| **165** | 60 | 110 | 145 | 300.0 |
| **166** | 60 | 115 | 145 | 310.2 |
| **167** | 75 | 120 | 150 | 320.4 |
| **168** | 75 | 125 | 150 | 330.4 |

169 rows × 4 columns

14.Replace the Nan entries in Net Sales with 0 using fillna()

df2['Calories'] = df2['Calories'].fillna(0)

df2

|  | **Duration** | **Pulse** | **Maxpulse** | **Calories** |
| --- | --- | --- | --- | --- |
| **0** | 60 | 110 | 130 | 409.1 |
| **1** | 60 | 117 | 145 | 479.0 |
| **2** | 60 | 103 | 135 | 340.0 |
| **3** | 45 | 109 | 175 | 282.4 |
| **4** | 45 | 117 | 148 | 406.0 |
| **...** | ... | ... | ... | ... |
| **164** | 60 | 105 | 140 | 290.8 |
| **165** | 60 | 110 | 145 | 300.0 |
| **166** | 60 | 115 | 145 | 310.2 |
| **167** | 75 | 120 | 150 | 320.4 |
| **168** | 75 | 125 | 150 | 330.4 |

169 rows × 4 columns

df2[:19]

|  | **Duration** | **Pulse** | **Maxpulse** | **Calories** |
| --- | --- | --- | --- | --- |
| **0** | 60 | 110 | 130 | 409.1 |
| **1** | 60 | 117 | 145 | 479.0 |
| **2** | 60 | 103 | 135 | 340.0 |
| **3** | 45 | 109 | 175 | 282.4 |
| **4** | 45 | 117 | 148 | 406.0 |
| **5** | 60 | 102 | 127 | 300.0 |
| **6** | 60 | 110 | 136 | 374.0 |
| **7** | 45 | 104 | 134 | 253.3 |
| **8** | 30 | 109 | 133 | 195.1 |
| **9** | 60 | 98 | 124 | 269.0 |
| **10** | 60 | 103 | 147 | 329.3 |
| **11** | 60 | 100 | 120 | 250.7 |
| **12** | 60 | 106 | 128 | 345.3 |
| **13** | 60 | 104 | 132 | 379.3 |
| **14** | 60 | 98 | 123 | 275.0 |
| **15** | 60 | 98 | 120 | 215.2 |
| **16** | 60 | 100 | 120 | 300.0 |
| **17** | 45 | 90 | 112 | 0.0 |
| **18** | 60 | 103 | 123 | 323.0 |

15.Replace entries which are 'MISSING' in order fufilled with False using replace()

**import** numpy **as** np

df2 **=** df2**.**replace(np**.**nan, **False**)

df2

|  | **Duration** | **Pulse** | **Maxpulse** | **Calories** |
| --- | --- | --- | --- | --- |
| **0** | 60 | 110 | 130 | 409.1 |
| **1** | 60 | 117 | 145 | 479.0 |
| **2** | 60 | 103 | 135 | 340.0 |
| **3** | 45 | 109 | 175 | 282.4 |
| **4** | 45 | 117 | 148 | 406.0 |
| **...** | ... | ... | ... | ... |
| **164** | 60 | 105 | 140 | 290.8 |
| **165** | 60 | 110 | 145 | 300.0 |
| **166** | 60 | 115 | 145 | 310.2 |
| **167** | 75 | 120 | 150 | 320.4 |
| **168** | 75 | 125 | 150 | 330.4 |

169 rows × 4 columns

16. Drop a column, axis=0 when dropping rows, axis=1 when dropping columns

df2.drop('Duration',axis=1)

|  | **Pulse** | **Maxpulse** | **Calories** |
| --- | --- | --- | --- |
| **0** | 110 | 130 | 409.1 |
| **1** | 117 | 145 | 479.0 |
| **2** | 103 | 135 | 340.0 |
| **3** | 109 | 175 | 282.4 |
| **4** | 117 | 148 | 406.0 |
| **...** | ... | ... | ... |
| **164** | 105 | 140 | 290.8 |
| **165** | 110 | 145 | 300.0 |
| **166** | 115 | 145 | 310.2 |
| **167** | 120 | 150 | 320.4 |
| **168** | 125 | 150 | 330.4 |

169 rows × 3 columns

df.drop(0,axis=0)

|  | **Invoice ID** | **Branch** | **City** | **Customer type** | **Gender** | **Product line** | **Unit price** | **Quantity** | **Tax 5%** | **Total** | **Date** | **Time** | **Payment** | **cogs** | **gross margin percentage** | **gross income** | **Rating** | **Rating\_Rank** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 226-31-3081 | C | Naypyitaw | Normal | Female | Electronic accessories | 15.28 | 5 | 3.8200 | 80.2200 | 3/8/2019 | 10:29 | Cash | 76.40 | 4.761905 | 3.8200 | 9.6 | 63.0 |
| **2** | 631-41-3108 | A | Yangon | Normal | Male | Home and lifestyle | 46.33 | 7 | 16.2155 | 340.5255 | 3/3/2019 | 13:23 | Credit card | 324.31 | 4.761905 | 16.2155 | 7.4 | 421.5 |
| **3** | 123-19-1176 | A | Yangon | Member | Male | Health and beauty | 58.22 | 8 | 23.2880 | 489.0480 | 1/27/2019 | 20:33 | Ewallet | 465.76 | 4.761905 | 23.2880 | 8.4 | 261.5 |
| **4** | 373-73-7910 | A | Yangon | Normal | Male | Sports and travel | 86.31 | 7 | 30.2085 | 634.3785 | 2/8/2019 | 10:37 | Ewallet | 604.17 | 4.761905 | 30.2085 | 5.3 | 785.0 |
| **5** | 699-14-3026 | C | Naypyitaw | Normal | Male | Electronic accessories | 85.39 | 7 | 29.8865 | 627.6165 | 3/25/2019 | 18:30 | Ewallet | 597.73 | 4.761905 | 29.8865 | 4.1 | 981.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **995** | 233-67-5758 | C | Naypyitaw | Normal | Male | Health and beauty | 40.35 | 1 | 2.0175 | 42.3675 | 1/29/2019 | 13:46 | Ewallet | 40.35 | 4.761905 | 2.0175 | 6.2 | 635.0 |
| **996** | 303-96-2227 | B | Mandalay | Normal | Female | Home and lifestyle | 97.38 | 10 | 48.6900 | 1022.4900 | 3/2/2019 | 17:16 | Ewallet | 973.80 | 4.761905 | 48.6900 | 4.4 | 924.0 |
| **997** | 727-02-1313 | A | Yangon | Member | Male | Food and beverages | 31.84 | 1 | 1.5920 | 33.4320 | 2/9/2019 | 13:22 | Cash | 31.84 | 4.761905 | 1.5920 | 7.7 | 371.5 |
| **998** | 347-56-2442 | A | Yangon | Normal | Male | Home and lifestyle | 65.82 | 1 | 3.2910 | 69.1110 | 2/22/2019 | 15:33 | Cash | 65.82 | 4.761905 | 3.2910 | 4.1 | 981.0 |
| **999** | 849-09-3807 | A | Yangon | Member | Female | Fashion accessories | 88.34 | 7 | 30.9190 | 649.2990 | 2/18/2019 | 13:28 | Cash | 618.38 | 4.761905 | 30.9190 | 6.6 | 560.5 |

999 rows × 18 columns

17.Drop a row if it has a certain value (in this case, 'TRUE')

data1=df[(df.Quantity==1)].index

df.drop(data1)

Out[77]:

|  | **Invoice ID** | **Branch** | **City** | **Customer type** | **Gender** | **Product line** | **Unit price** | **Quantity** | **Tax 5%** | **Total** | **Date** | **Time** | **Payment** | **cogs** | **gross margin percentage** | **gross income** | **Rating** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 750-67-8428 | A | Yangon | Member | Female | Health and beauty | 74.69 | 7 | 26.1415 | 548 | 1/5/2019 | 13:08 | Ewallet | 522.83 | 4.761905 | 26.1415 | 9.1 |
| **1** | 226-31-3081 | C | Naypyitaw | Normal | Female | Electronic accessories | 15.28 | 5 | 3.8200 | 80 | 3/8/2019 | 10:29 | Cash | 76.40 | 4.761905 | 3.8200 | 9.6 |
| **2** | 631-41-3108 | A | Yangon | Normal | Male | Home and lifestyle | 46.33 | 7 | 16.2155 | 340 | 3/3/2019 | 13:23 | Credit card | 324.31 | 4.761905 | 16.2155 | 7.4 |
| **3** | 123-19-1176 | A | Yangon | Member | Male | Health and beauty | 58.22 | 8 | 23.2880 | 489 | 1/27/2019 | 20:33 | Ewallet | 465.76 | 4.761905 | 23.2880 | 8.4 |
| **4** | 373-73-7910 | A | Yangon | Normal | Male | Sports and travel | 86.31 | 7 | 30.2085 | 634 | 2/8/2019 | 10:37 | Ewallet | 604.17 | 4.761905 | 30.2085 | 5.3 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **991** | 602-16-6955 | B | Mandalay | Normal | Female | Sports and travel | 76.60 | 10 | 38.3000 | 804 | 1/24/2019 | 18:10 | Ewallet | 766.00 | 4.761905 | 38.3000 | 6.0 |
| **992** | 745-74-0715 | A | Yangon | Normal | Male | Electronic accessories | 58.03 | 2 | 5.8030 | 121 | 3/10/2019 | 20:46 | Ewallet | 116.06 | 4.761905 | 5.8030 | 8.8 |
| **993** | 690-01-6631 | B | Mandalay | Normal | Male | Fashion accessories | 17.49 | 10 | 8.7450 | 183 | 2/22/2019 | 18:35 | Ewallet | 174.90 | 4.761905 | 8.7450 | 6.6 |
| **996** | 303-96-2227 | B | Mandalay | Normal | Female | Home and lifestyle | 97.38 | 10 | 48.6900 | 1022 | 3/2/2019 | 17:16 | Ewallet | 973.80 | 4.761905 | 48.6900 | 4.4 |
| **999** | 849-09-3807 | A | Yangon | Member | Female | Fashion accessories | 88.34 | 7 | 30.9190 | 649 | 2/18/2019 | 13:28 | Cash | 618.38 | 4.761905 | 30.9190 | 6.6 |

888 rows × 17 columns