```
## DMW A1: Data Preprocessing
## Aditya Agre TYCOA6

## Data set : movies.csv
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('business-financial-data-june-2023-quarter-csv.csv')
df
```

	Series_reference	Period	Data_value	Suppressed	STATUS	UNITS	Magnit
0	BDCQ.SF1AA2CA	2016.06	1116.386	NaN	F	Dollars	
1	BDCQ.SF1AA2CA	2016.09	1070.874	NaN	F	Dollars	
2	BDCQ.SF1AA2CA	2016.12	1054.408	NaN	F	Dollars	
3	BDCQ.SF1AA2CA	2017.03	1010.665	NaN	F	Dollars	
4	BDCQ.SF1AA2CA	2017.06	1233.700	NaN	F	Dollars	

...	...	...	...	...	...	...
6910	BDCQ.SF8RSCA	2022.06	579.955	NaN	F	Dollars
6911	BDCQ.SF8RSCA	2022.09	609.161	NaN	F	Dollars
6912	BDCQ.SF8RSCA	2022.12	518.615	NaN	F	Dollars
6913	BDCQ.SF8RSCA	2023.03	663.630	NaN	F	Dollars
6914	BDCQ.SF8RSCA	2023.06	617.507	NaN	F	Dollars

6915 rows × 14 columns

```
## 5) Going through all measures of central tendency
## Studying modal values per column
df.mode(axis = 0)
```

	Series_reference	Period	Data_value	Suppressed	STATUS	UNITS	Magnitu
0	BDCQ.SF1AA2CA	2017.09	847.349	Y	F	Dollars	
1	BDCQ.SF1AA2CS	2017.12	925.547	NaN	NaN	NaN	N
2	BDCQ.SF1AA2CT	2018.03	1040.500	NaN	NaN	NaN	N
3	BDCQ.SF1AA3CA	2018.06	1063.775	NaN	NaN	NaN	N
4	BDCQ.SF1AA3CS	2018.09	1077.447	NaN	NaN	NaN	N
...	...	...	...	...	...	...	
226	BDCQ.SF8QQ1CA	NaN	NaN	NaN	NaN	NaN	N
227	BDCQ.SF8QQCA	NaN	NaN	NaN	NaN	NaN	N
228	BDCQ.SF8RS1CA	NaN	NaN	NaN	NaN	NaN	N
229	BDCQ.SF8RS2CA	NaN	NaN	NaN	NaN	NaN	N
230	BDCQ.SF8RSCA	NaN	NaN	NaN	NaN	NaN	N

231 rows × 14 columns

```
## Studying mean values per column
df.mean()
```

```
/var/folders/ft/m0h88bl55gl0qmgjxz9qczfc0000gn/T/ipykernel_55282/2762134590
  df.mean()
Period            2019.473197
Data_value        4826.080308
Magnitude            6.000000
Series_title_5            NaN
dtype: float64
```

```
## Studying median values per column
df.median()
```

```
    /var/folders/ft/m0h88bl55gl0qmgjxz9qczfc0000gn/T/ipykernel_55282/1669262333
      df.median()
    Period           2019.1200
    Data_value       2118.3655
    Magnitude           6.0000
    Series_title_5         NaN
    dtype: float64
```

```
## Handling Missing Values
```

```
## which of these columns have null values

no_of_col = df.shape[1]

for i in range(no_of_col):
    if(df[:i].isnull().values.any()):
        print("Col ",i," has null values.")
```

```
    Col  1  has null values.
    Col  2  has null values.
    Col  3  has null values.
    Col  4  has null values.
    Col  5  has null values.
    Col  6  has null values.
    Col  7  has null values.
    Col  8  has null values.
    Col  9  has null values.
    Col  10  has null values.
    Col  11  has null values.
    Col  12  has null values.
    Col  13  has null values.
```

```
## Checking number of null entries per column.
df.isnull().sum()
```

```
Series_reference       0
Period                 0
Data_value           605
Suppressed          6870
STATUS                 0
UNITS                  0
Magnitude              0
Subject                0
Group                  0
Series_title_1         0
Series_title_2         0
Series_title_3         0
Series_title_4         0
Series_title_5      6915
dtype: int64
```

```
null_col = []
for col in df:

    ## print(col)
    null_col.append(col)

null_col

for col in null_col:
    df[col] = df[col].fillna(df[col].mode())
df
```

	Series_reference	Period	Data_value	Suppressed	STATUS	UNITS	Magnit
0	BDCQ.SF1AA2CA	2016.06	1116.386	Y	F	Dollars	
1	BDCQ.SF1AA2CA	2016.09	1070.874	NaN	F	Dollars	
2	BDCQ.SF1AA2CA	2016.12	1054.408	NaN	F	Dollars	

```
   3  BDCQ.SF1AA2CA  2017.03  1010.665  NaN  F  Dollars

   4  BDCQ.SF1AA2CA  2017.06  1233.700  NaN  F  Dollars

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

6910   BDCQ.SF8RSCA  2022.06   579.955  NaN  F  Dollars

6911   BDCQ.SF8RSCA  2022.09   609.161  NaN  F  Dollars

6912   BDCQ.SF8RSCA  2022.12   518.615  NaN  F  Dollars

6913   BDCQ.SF8RSCA  2023.03   663.630  NaN  F  Dollars

6914   BDCQ.SF8RSCA  2023.06   617.507  NaN  F  Dollars

6915 rows × 14 columns
```

```
## Checking number of null entries per column.
df.isnull().sum()
```

```
Series_reference       0
Period                 0
Data_value           591
Suppressed          6869
STATUS                 0
UNITS                  0
Magnitude              0
Subject                0
Group                  0
Series_title_1         0
Series_title_2         0
Series_title_3         0
Series_title_4         0
Series_title_5      6915
dtype: int64
```

```
df
```

	Series_reference	Period	Data_value	Suppressed	STATUS	UNITS	Magnit
0	BDCQ.SF1AA2CA	2016.06	1116.386	Y	F	Dollars	
1	BDCQ.SF1AA2CA	2016.09	1070.874	NaN	F	Dollars	
2	BDCQ.SF1AA2CA	2016.12	1054.408	NaN	F	Dollars	
3	BDCQ.SF1AA2CA	2017.03	1010.665	NaN	F	Dollars	

```
   4  BDCQ.SF1AA2CA  2017.06  1233.700  NaN  F  Dollars
 ...            ...      ...       ...  ...  ...    ...
6910   BDCQ.SF8RSCA  2022.06   579.955  NaN  F  Dollars
6911   BDCQ.SF8RSCA  2022.09   609.161  NaN  F  Dollars
6912   BDCQ.SF8RSCA  2022.12   518.615  NaN  F  Dollars
6913   BDCQ.SF8RSCA  2023.03   663.630  NaN  F  Dollars
6914   BDCQ.SF8RSCA  2023.06   617.507  NaN  F  Dollars

6915 rows × 14 columns
```

```
df['Data_value'].replace(np.NaN, df['Data_value'].mode()[0],inplace=True)
df
```

```
     Series_reference  Period  Data_value  Suppressed  STATUS  UNITS  Magnit
```

```
   0  BDCQ.SF1AA2CA  2016.06  1116.386    Y  F  Dollars

   1  BDCQ.SF1AA2CA  2016.09  1070.874  NaN  F  Dollars

   2  BDCQ.SF1AA2CA  2016.12  1054.408  NaN  F  Dollars

   3  BDCQ.SF1AA2CA  2017.03  1010.665  NaN  F  Dollars

   4  BDCQ.SF1AA2CA  2017.06  1233.700  NaN  F  Dollars

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

6910    BDCQ.SF8RSCA  2022.06   579.955  NaN  F  Dollars

6911    BDCQ.SF8RSCA  2022.09   609.161  NaN  F  Dollars

6912    BDCQ.SF8RSCA  2022.12   518.615  NaN  F  Dollars
```

6913	BDCQ.SF8RSCA	2023.03	663.630	NaN	F	Dollars
6914	BDCQ.SF8RSCA	2023.06	617.507	NaN	F	Dollars

6915 rows × 14 columns

```
# Lets look at the duplicate rows now
df[df.duplicated(keep='first')]
# Removing these rows
df.drop_duplicates(keep='first', inplace=True)
df
```

	Series_reference	Period	Data_value	Suppressed	STATUS	UNITS	Magnit
0	BDCQ.SF1AA2CA	2016.06	1116.386	Y	F	Dollars	
1	BDCQ.SF1AA2CA	2016.09	1070.874	NaN	F	Dollars	
2	BDCQ.SF1AA2CA	2016.12	1054.408	NaN	F	Dollars	
3	BDCQ.SF1AA2CA	2017.03	1010.665	NaN	F	Dollars	

4	BDCQ.SF1AA2CA	2017.06	1233.700	NaN	F	Dollars
...	...	...	...	...	...	...
6910	BDCQ.SF8RSCA	2022.06	579.955	NaN	F	Dollars
6911	BDCQ.SF8RSCA	2022.09	609.161	NaN	F	Dollars
6912	BDCQ.SF8RSCA	2022.12	518.615	NaN	F	Dollars
6913	BDCQ.SF8RSCA	2023.03	663.630	NaN	F	Dollars
6914	BDCQ.SF8RSCA	2023.06	617.507	NaN	F	Dollars

6915 rows × 14 columns

```
## data has not been arranged according to any pattern
## Yet, we are shuffling

df = df.sample(frac=1, random_state=42)
```

```
df
```

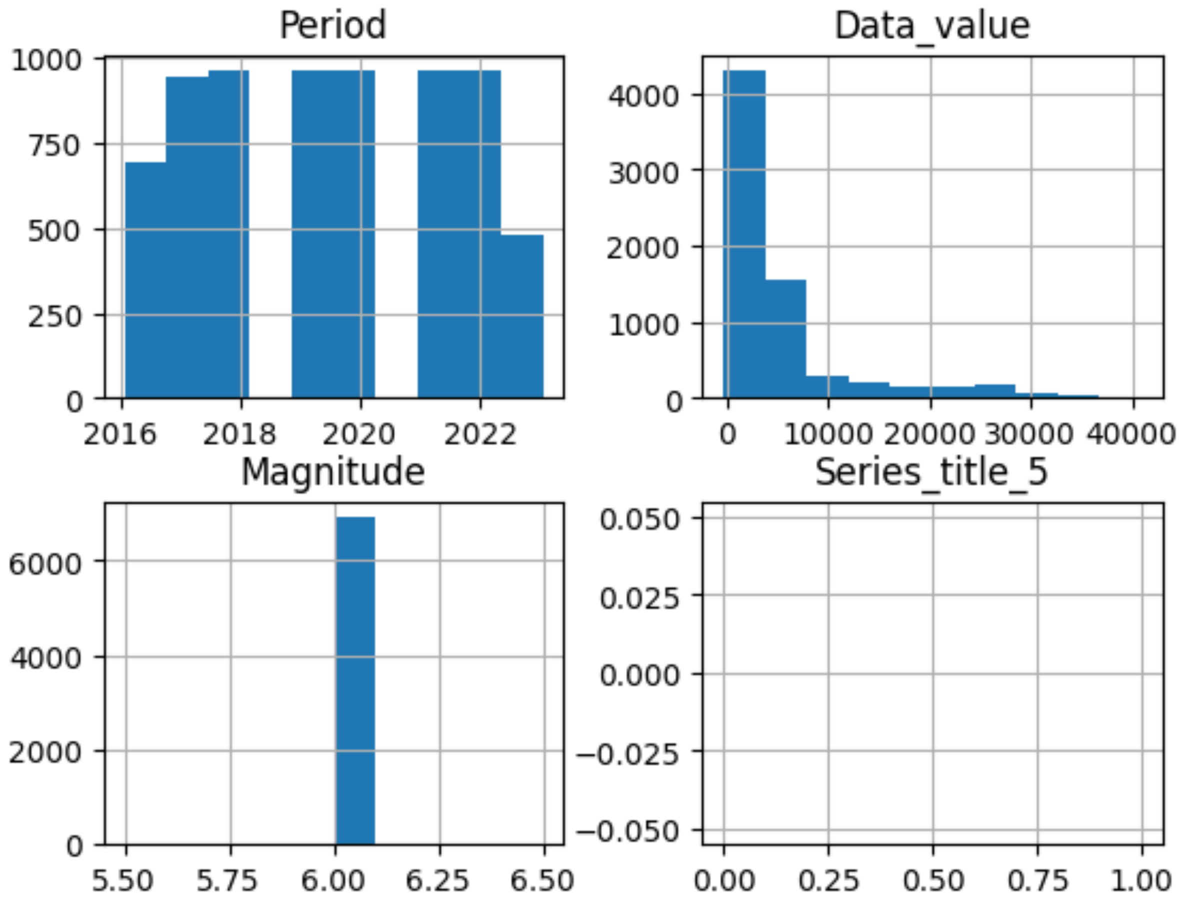
	Series_reference	Period	Data_value	Suppressed	STATUS	UNITS	Magnit
5219	BDCQ.SF3FFCA	2019.12	2069.680	NaN	F	Dollars	
828	BDCQ.SF1CC5CS	2020.06	3413.355	NaN	R	Dollars	
3269	BDCQ.SF1RS1CS	2021.09	1299.285	NaN	R	Dollars	
1433	BDCQ.SF1DDCS	2019.06	5296.602	NaN	R	Dollars	
4937	BDCQ.SF3CC6CA	2021.12	201.206	NaN	F	Dollars	
...	...	...	...	...	...	...	
3772	BDCQ.SF2CC6CA	2016.12	620.816	NaN	F	Dollars	
5191	BDCQ.SF3FF1CA	2020.03	1954.248	NaN	F	Dollars	

5226	BDCQ.SF3FFCA	2021.09	2165.344	NaN	F	Dollars
5390	BDCQ.SF3JJ1CA	2022.12	688.216	NaN	F	Dollars
860	BDCQ.SF1CC5CT	2021.03	5165.815	NaN	C	Dollars

6915 rows × 14 columns

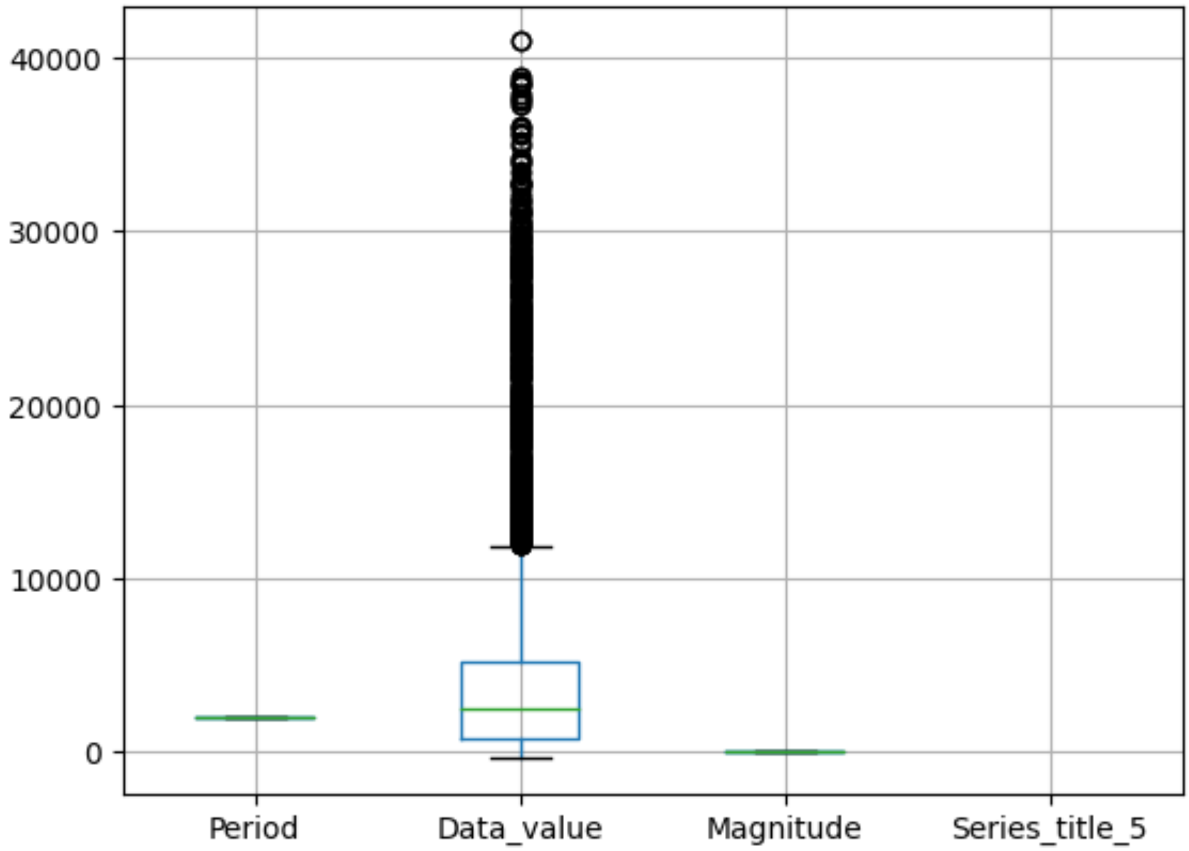
```
## Normalising Data using min max scaling
## PLotting histogram

df.hist()
```

```
array([[<AxesSubplot:title={'center':'Period'}>,
        <AxesSubplot:title={'center':'Data_value'}>],
       [<AxesSubplot:title={'center':'Magnitude'}>,
        <AxesSubplot:title={'center':'Series_title_5'}>]], dtype=object)
```



```
## Plotting boxplot

df.boxplot()
```

```
<AxesSubplot:>
```



```
# Boxplot shows many outliers

# lets make a general func that removes outliers for a given column

def outlier_remove(col_i):
    ## Lets follow inter quartile range method
    ## return the values of the lower range limit and upper range limit
    ## these limits have values 1.5*(inter quartile range beyond) first and thir

    ## To find quartiles, we must sort the column
    sorted(col_i)
    print(col_i)

    Q1,Q3 = np.percentile(col_i , [25,75])
    ## because we are taking quartiles. therefore 25% and 75%
    inter_q_range = Q3-Q1

    l_lim = Q1 - (1.5 * inter_q_range)
    up_lim = Q3 + (1.5 * inter_q_range)
    return l_lim,up_lim
```

```
l,u=outlier_remove(df.Data_value)
print(l,u,q1,q3,iqr)

rows = df.shape[0]

df.drop(df[(df.Data_value < l) | (df.Data_value > u)].index,inplace=True)
```

```
5219    2069.680
828     3413.355
3269    1299.285
1433    5296.602
4937     201.206
          ...
3772     620.816
5191    1954.248
5226    2165.344
5390     688.216
860     5165.815
Name: Data_value, Length: 6915, dtype: float64
-5928.416249999998 11822.353749999998 nan nan nan
```

```
l,u=outlier_remove(df.Data_value)
print(l,u,q1,q3,iqr)

rows = df.shape[0]

df.drop(df[(df.Data_value < l) | (df.Data_value > u)].index,inplace=True)
```

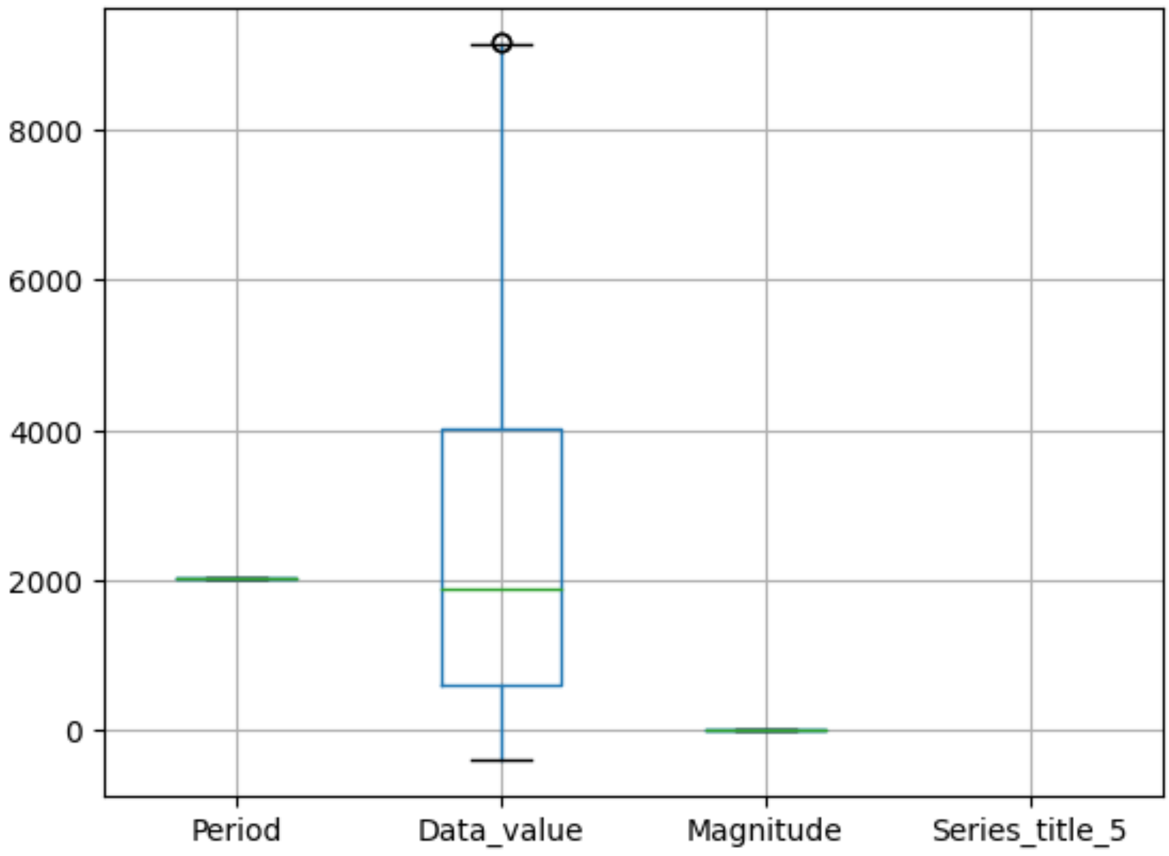
```
5219    2069.680
828     3413.355
3269    1299.285
1433    5296.602
4937     201.206
          ...
3772     620.816
5191    1954.248
5226    2165.344
5390     688.216
860     5165.815
Name: Data_value, Length: 6108, dtype: float64
-4733.8478749999995 9570.571124999999 nan nan nan
```

```
l,u=outlier_remove(df.Data_value)
print(l,u,q1,q3,iqr)

rows = df.shape[0]

df.drop(df[(df.Data_value < l) | (df.Data_value > u)].index,inplace=True)
```

```
5219    2069.680
828     3413.355
3269    1299.285
1433    5296.602
4937     201.206
          ...
3772     620.816
5191    1954.248
5226    2165.344
5390     688.216
860     5165.815
Name: Data_value, Length: 6003, dtype: float64
-4514.4527499999995 9174.881249999999 nan nan nan
```

```
df.boxplot()
```

```
<AxesSubplot:>
```



```
## Outliers removed
```

```
from sklearn.preprocessing import MinMaxScaler

MMscaler = MinMaxScaler()
## We have created an object of minmax scaler class
```

```
df_ncol= df.select_dtypes(exclude=['object'])
df_ncol
## We are doing this to extract onlythe numeric cols
## A new dataframe is being created with just these cols
```

	Period	Data_value	Magnitude	Series_title_5
5219	0.437143	0.258364	0.0	NaN
828	0.571429	0.399034	0.0	NaN
3269	0.718571	0.177710	0.0	NaN
1433	0.428571	0.596193	0.0	NaN
4937	0.722857	0.062752	0.0	NaN
...	...	...	...	...
3772	0.008571	0.106681	0.0	NaN
5191	0.567143	0.246279	0.0	NaN
5226	0.718571	0.268379	0.0	NaN
5390	0.865714	0.113737	0.0	NaN
860	0.710000	0.582500	0.0	NaN

5986 rows × 4 columns

```
## Making a copy of df
temp = df

# Scalable columns
cols= df_ncol.columns

## Performing min max scalning
temp[cols]=  MMscaler.fit_transform(df[cols])
```

```
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packag
  data_min = np.nanmin(X, axis=0)
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packag
  data_max = np.nanmax(X, axis=0)
```

```
temp
```

	Series_reference	Period	Data_value	Suppressed	STATUS	UNITS	Magni

```
5219  BDCQ.SF3FFCA  0.437143  0.258364  NaN  F  Dollars

 828  BDCQ.SF1CC5CS 0.571429  0.399034  NaN  R  Dollars

3269  BDCQ.SF1RS1CS 0.718571  0.177710  NaN  R  Dollars

1433  BDCQ.SF1DDCS  0.428571  0.596193  NaN  R  Dollars

4937  BDCQ.SF3CC6CA 0.722857  0.062752  NaN  F  Dollars

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

3772  BDCQ.SF2CC6CA 0.008571  0.106681  NaN  F  Dollars

5191  BDCQ.SF3FF1CA 0.567143  0.246279  NaN  F  Dollars

5226  BDCQ.SF3FFCA  0.718571  0.268379  NaN  F  Dollars
```

```
5390   BDCQ.SF3JJ1CA   0.865714   0.113737        NaN     F   Dollars

860    BDCQ.SF1CC5CT   0.710000   0.582500        NaN     C   Dollars

5986 rows × 14 columns
```

```
## Columns scaled using min max scaling
```