Importing Libraries and Data

In [ ]:
```
#Shubhendra Kumar
#Email: kshubhendra8860@gmail.com
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_excel(r"DS_Python_Assignment.xlsx", sheet_name='Data')
df
```

Out[2]:

	CUSTOMERID	STATE	LCPCOUNT	PRIVATELABELTENDERFLAG	TENURE_IN_MONTHS	CLOSESTSTOREDISTANCE	FEMALE	AGE	HS_DIPLOMA	SOME
0	5001	TX	1	N	-9.0	NaN	0	NaN	0	
1	5002	OH	0	Y	9.0	8.728943	0	NaN	0	
2	5003	TX	0	N	12.0	NaN	0	NaN	0	
3	5004	TN	0	N	-1.0	NaN	0	NaN	0	
4	5005	TX	0	N	16.0	NaN	0	NaN	0	
...	...	...	...	...	...	...	...	...	...	
9995	14996	PA	1	Y	139.0	37.827206	1	41.0	0	
9996	14997	PA	1	N	28.0	40.064053	0	49.0	0	
9997	14998	DC	2	N	10.0	2.280174	0	NaN	0	
9998	14999	CO	1	N	23.0	6.926766	0	NaN	0	
9999	15000	TX	1	N	229.0	18.602889	0	NaN	0	

10000 rows × 117 columns

Exploring the data

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Columns: 117 entries, CUSTOMERID to MDAYSHOPPER_L4Y
dtypes: float64(49), int64(65), object(3)
memory usage: 8.9+ MB
```

In [4]:
```
df.describe()
```

Out[4]:

	CUSTOMERID	LCPCOUNT	TENURE_IN_MONTHS	CLOSESTSTOREDISTANCE	FEMALE	AGE	HS_DIPLOMA	SOME_COLLEGE	BACH_GRA
count	10000.00000	10000.000000	9934.000000	8455.000000	10000.000000	4055.000000	10000.00000	10000.000000	10000
mean	10000.50000	1.171900	54.286290	21.126264	0.156700	47.351418	0.09840	0.133800	0
std	2886.89568	2.067118	70.087569	83.097443	0.363536	15.350262	0.29787	0.340454	0
min	5001.00000	0.000000	-9.000000	0.000000	0.000000	19.000000	0.00000	0.000000	0
25%	7500.75000	0.000000	10.000000	4.643872	0.000000	35.000000	0.00000	0.000000	0
50%	10000.50000	1.000000	28.000000	8.442909	0.000000	47.000000	0.00000	0.000000	0
75%	12500.25000	1.000000	62.000000	17.793590	0.000000	59.000000	0.00000	0.000000	0
max	15000.00000	75.000000	287.000000	4763.484736	1.000000	87.000000	1.00000	1.000000	1

8 rows × 114 columns

Working on Null Values

In [5]:
```
#Find the null values
df.isnull()
```

Out[5]:

	CUSTOMERID	STATE	LCPCOUNT	PRIVATELABELTENDERFLAG	TENURE_IN_MONTHS	CLOSESTSTOREDISTANCE	FEMALE	AGE	HS_DIPLOMA	SOMI
0	False	False	False	False	False	True	False	True	False	
1	False	False	False	False	False	False	False	True	False	
2	False	False	False	False	False	True	False	True	False	
3	False	False	False	False	False	True	False	True	False	
4	False	False	False	False	False	True	False	True	False	
...	...	...	...	...	...	...	...	...	...	
9995	False	False	False	False	False	False	False	False	False	
9996	False	False	False	False	False	False	False	False	False	
9997	False	False	False	False	False	False	False	True	False	
9998	False	False	False	False	False	False	False	True	False	
9999	False	False	False	False	False	False	False	True	False	

10000 rows × 117 columns

In [6]:
```
#Number of null values in first 20 columns.
df.isnull().sum().head(20)
```

Out[6]:
```
CUSTOMERID                 0
STATE                     59
LCPCOUNT                   0
PRIVATELABELTENDERFLAG     0
TENURE_IN_MONTHS          66
CLOSESTSTOREDISTANCE    1545
FEMALE                     0
AGE                     5945
HS_DIPLOMA                 0
SOME_COLLEGE               0
BACH_GRAD_DEG              0
LT_HS_DIPLOMA              0
MARRIED                    0
MNGMNT_OFFICEADMIN         0
TECH_PROF                  0
SALES_JOB                  0
BLUE_COLLAR                0
FARMER                     0
RETIRED                    0
SFDU                       0
dtype: int64
```

In [7]:
```
#Number of null values in last 20 columns.
df.isnull().sum().tail(20)
```

Out[7]:
```
7-Cost Only Spend                0
8-Marketing Premium SKUs Spend   0
9-Repairs & Appraisals Spend     0
10-Pre Owned Spend               0
11-Watches Spend                 0
12-Misc Merchandise Spend        0
15-Store Events Spend            0
16-Single Stone Jewelry Spend    0
MDAYREV_L1Y                      0
MDAYREV_L2Y                      0
MDAYREV_L3Y                      0
MDAYREV_L4Y                      0
MDAYQTY_L1Y                      0
MDAYQTY_L2Y                      0
MDAYQTY_L3Y                      0
MDAYQTY_L4Y                      0
MDAYSHOPPER_L1Y                  0
MDAYSHOPPER_L2Y                  0
MDAYSHOPPER_L3Y                  0
MDAYSHOPPER_L4Y                  0
dtype: int64
```

In [8]:
```
#Finding the number with any null value in the columns.
df.columns[df.isnull().any()]
```

Out[8]:
```
Index(['STATE', 'TENURE_IN_MONTHS', 'CLOSESTSTOREDISTANCE', 'AGE', 'INCOME',
       'LENGTH OF RESIDENCE', 'NUMBER OF PERSONS IN LIVING UNIT',
       'NUMBER OF ADULTS IN LIVING UNIT', 'MOSAIC',
       'CAPE: AGE: POP: MEDIAN AGE', 'CAPE: AGE: POP: % 0-17',
       'CAPE: AGE: POP: % 18-99+', 'CAPE: AGE: POP: % 65-99+',
       'CAPE: ETHNIC: POP: % WHITE ONLY', 'CAPE: ETHNIC: POP: % BLACK ONLY',
       'CAPE: ETHNIC: POP: % ASIAN ONLY', 'CAPE: ETHNIC: POP: % HISPANIC',
       'CAPE: DENSITY: PERSONS PER HH FOR POP IN HH',
       'CAPE: HHSIZE: HH: AVERAGE HOUSEHOLD SIZE',
       'CAPE: TYP: HH: % MARRIED COUPLE FAMILY',
       'CAPE: CHILD: HH: % WITH PERSONS LT18',
       'CAPE: CHILD: HH: % MARR COUPLE FAMW- PERSONS LT18',
       'CAPE: CHILD: HH: % MARR COUPLE FAMW-O PERSONS LT18',
       'CAPE: LANG: HH: % SPANISH SPEAKING',
       'CAPE: EDUC: POP25+: MEDIAN EDUCATION ATTAINED',
       'CAPE: HOMVAL: OOHU: MEDIAN HOME VALUE',
       'CAPE: BUILT: HU: MEDIAN HOUSING UNIT AGE',
       'CAPE: TENANCY: OCCHU: % OWNER OCCUPIED',
       'CAPE: TENANCY: OCCHU: % RENTER OCCUPIED', 'CAPE: EDUC: ISPSA',
       'CAPE: EDUC: ISPSA DECILE', 'CAPE: INC: FAMILY INC STATE DECILE',
       'CAPE: INC: HH: MEDIAN FAMILY HOUSEHOLD INCOME'],
      dtype='object')
```

List down all the columns with missing values.

In [9]:
```
#List of all the null value in the columns.
df.columns[df.isnull().any()].tolist()
```

Out[9]:
```
['STATE',
 'TENURE_IN_MONTHS',
 'CLOSESTSTOREDISTANCE',
 'AGE',
 'INCOME',
 'LENGTH OF RESIDENCE',
 'NUMBER OF PERSONS IN LIVING UNIT',
 'NUMBER OF ADULTS IN LIVING UNIT',
 'MOSAIC',
 'CAPE: AGE: POP: MEDIAN AGE',
 'CAPE: AGE: POP: % 0-17',
 'CAPE: AGE: POP: % 18-99+',
 'CAPE: AGE: POP: % 65-99+',
 'CAPE: ETHNIC: POP: % WHITE ONLY',
 'CAPE: ETHNIC: POP: % BLACK ONLY',
 'CAPE: ETHNIC: POP: % ASIAN ONLY',
 'CAPE: ETHNIC: POP: % HISPANIC',
 'CAPE: DENSITY: PERSONS PER HH FOR POP IN HH',
 'CAPE: HHSIZE: HH: AVERAGE HOUSEHOLD SIZE',
 'CAPE: TYP: HH: % MARRIED COUPLE FAMILY',
 'CAPE: CHILD: HH: % WITH PERSONS LT18',
 'CAPE: CHILD: HH: % MARR COUPLE FAMW- PERSONS LT18',
 'CAPE: CHILD: HH: % MARR COUPLE FAMW-O PERSONS LT18',
 'CAPE: LANG: HH: % SPANISH SPEAKING',
 'CAPE: EDUC: POP25+: MEDIAN EDUCATION ATTAINED',
 'CAPE: HOMVAL: OOHU: MEDIAN HOME VALUE',
 'CAPE: BUILT: HU: MEDIAN HOUSING UNIT AGE',
 'CAPE: TENANCY: OCCHU: % OWNER OCCUPIED',
 'CAPE: TENANCY: OCCHU: % RENTER OCCUPIED',
 'CAPE: EDUC: ISPSA',
 'CAPE: EDUC: ISPSA DECILE',
 'CAPE: INC: FAMILY INC STATE DECILE',
 'CAPE: INC: HH: MEDIAN FAMILY HOUSEHOLD INCOME']
```

In [10]:
```
#Exploring the data type of different columns
df.dtypes
```

Out[10]:
```
CUSTOMERID                 int64
STATE                     object
LCPCOUNT                   int64
PRIVATELABELTENDERFLAG    object
TENURE_IN_MONTHS         float64
                           ...
MDAYQTY_L4Y                int64
MDAYSHOPPER_L1Y            int64
MDAYSHOPPER_L2Y            int64
MDAYSHOPPER_L3Y            int64
MDAYSHOPPER_L4Y            int64
Length: 117, dtype: object
```

In [11]:
```
#Checking for unique values of data type
df.dtypes.unique()
```

Out[11]:
```
array([dtype('int64'), dtype('O'), dtype('float64')], dtype=object)
```

In [12]:
```
#Exploring the data type of head.
df.dtypes.head()
```

Out[12]:
```
CUSTOMERID                 int64
STATE                     object
LCPCOUNT                   int64
PRIVATELABELTENDERFLAG    object
TENURE_IN_MONTHS         float64
dtype: object
```

In [13]:
```
#Exploring the data type of tail.
df.dtypes.tail()
```

Out[13]:
```
MDAYQTY_L4Y        int64
MDAYSHOPPER_L1Y    int64
MDAYSHOPPER_L2Y    int64
MDAYSHOPPER_L3Y    int64
MDAYSHOPPER_L4Y    int64
dtype: object
```

In [14]:
```
#printing the Integer data type.
int_columns = df.select_dtypes(include='int64').columns.tolist()
print("Int64 Columns:",int_columns)
```

```
Int64 Columns: ['CUSTOMERID', 'LCPCOUNT', 'FEMALE', 'HS_DIPLOMA', 'SOME_COLLEGE', 'BACH_GRAD_DEG', 'LT_HS_DIPLOMA', 'MARRIED',
'MNGMNT_OFFICEADMIN', 'TECH_PROF', 'SALES_JOB', 'BLUE_COLLAR', 'FARMER', 'RETIRED', 'SFDU', 'MFDU', 'HOMEOWNER', 'MAIL_RESP_MULT
I', 'MAIL_RESP_SINGLE', 'METRO', 'URBAN', 'MOR BANK: UPSCALE MERCHANDISE BUYER', 'MOR BANK: MALE MERCHANDISE BUYER', 'MOR BANK:
FEMALE MERCHANDISE BUYER', 'MOR BANK: CRAFTS-HOBBY MERCHANDISE BUYER', 'MOR BANK: GARDENING-FARMING BUYER', 'MOR BANK: BOOK BUYE
R', 'MOR BANK: COLLECT-SPECIAL FOODS BUYER', 'MOR BANK: GIFTS AND GADGETS BUYER', 'MOR BANK: GENERAL MERCHANDISE BUYER', 'MOR BA
NK: FAMILY AND GENERAL MAGAZINE', 'MOR BANK: FEMALE ORIENTED MAGAZINE', 'MOR BANK: MALE SPORTS MAGAZINE', 'MOR BANK: RELIGIOUS M
AGAZINE', 'MOR BANK: GARDENING-FARMING MAGAZINE', 'MOR BANK: CULINARY INTERESTS MAGAZINE', 'MOR BANK: HEALTH AND FITNESS MAGAZIN
E', 'MOR BANK: DO-IT-YOURSELFERS', 'MOR BANK: NEWS AND FINANCIAL', 'MOR BANK: PHOTOGRAPHY', 'MOR BANK: OPPORTUNITY SEEKERS AND C
E', 'MOR BANK: RELIGIOUS CONTRIBUTOR', 'MOR BANK: POLITICAL CONTRIBUTOR', 'MOR BANK: HEALTH AND INSTITUTION CONTRIBUTOR', 'MOR B
ANK: GENERAL CONTRIBUTOR', 'MOR BANK: MISCELLANEOUS', 'MOR BANK: ODDS AND ENDS', 'MOR BANK: DEDUPED CATEGORY HIT COUNT', 'MOR BA
NK: NON-DEDUPED CATEGORY HIT COUNT', 'MORTGAGE-HOME PURCHASE: HOME PURCHASE PRICE', 'CHILDREN', 'FREQUENCY', 'QUANTITY', 'FREQUE
NCY_2Y', 'QUANTITY_2Y', '7-Cost Only Spend', '9-Repairs & Appraisals Spend', 'MDAYQTY_L1Y', 'MDAYQTY_L2Y', 'MDAYQTY_L3Y', 'MDAYQ
TY_L4Y', 'MDAYSHOPPER_L1Y', 'MDAYSHOPPER_L2Y', 'MDAYSHOPPER_L3Y', 'MDAYSHOPPER_L4Y']
```

In [15]:
```
#printing the Float data type.
float_columns = df.select_dtypes(include='float64').columns.tolist()
print("Columns with float64: ",float_columns)
```

```
Columns with float64:  ['TENURE_IN_MONTHS', 'CLOSESTSTOREDISTANCE', 'AGE', 'INCOME', 'LENGTH OF RESIDENCE', 'NUMBER OF PERSONS I
N LIVING UNIT', 'NUMBER OF ADULTS IN LIVING UNIT', 'CAPE: AGE: POP: MEDIAN AGE', 'CAPE: AGE: POP: % 0-17', 'CAPE: AGE: POP: % 18
-99+', 'CAPE: AGE: POP: % 65-99+', 'CAPE: ETHNIC: POP: % WHITE ONLY', 'CAPE: ETHNIC: POP: % BLACK ONLY', 'CAPE: ETHNIC: POP: % A
SIAN ONLY', 'CAPE: ETHNIC: POP: % HISPANIC', 'CAPE: DENSITY: PERSONS PER HH FOR POP IN HH', 'CAPE: HHSIZE: HH: AVERAGE HOUSEHOLD
SIZE', 'CAPE: TYP: HH: % MARRIED COUPLE FAMILY', 'CAPE: CHILD: HH: % WITH PERSONS LT18', 'CAPE: CHILD: HH: % MARR COUPLE FAMW- P
ERSONS LT18', 'CAPE: CHILD: HH: % MARR COUPLE FAMW-O PERSONS LT18', 'CAPE: LANG: HH: % SPANISH SPEAKING', 'CAPE: EDUC: POP25+: M
EDIAN EDUCATION ATTAINED', 'CAPE: HOMVAL: OOHU: MEDIAN HOME VALUE', 'CAPE: BUILT: HU: MEDIAN HOUSING UNIT AGE', 'CAPE: TENANCY:
OCCHU: % OWNER OCCUPIED', 'CAPE: TENANCY: OCCHU: % RENTER OCCUPIED', 'CAPE: EDUC: ISPSA', 'CAPE: EDUC: ISPSA DECILE', 'CAPE: IN
C: FAMILY INC STATE DECILE', 'CAPE: INC: HH: MEDIAN FAMILY HOUSEHOLD INCOME', 'TOTALSALES', 'TOTALSALES_2Y', '1-Engagement Spen
d', '2-Wedding Bands Spend', '3-Fashion Diamonds Spend', '4-Fashion Jewelry Spend', '5-Close Out Spend', '6-Promotional Items Sp
end', '8-Marketing Premium SKUs Spend', '10-Pre Owned Spend', '11-Watches Spend', '12-Misc Merchandise Spend', '15-Store Events
Spend', '16-Single Stone Jewelry Spend', 'MDAYREV_L1Y', 'MDAYREV_L2Y', 'MDAYREV_L3Y', 'MDAYREV_L4Y']
```

In [16]:
```
#printing the Object data type.
object_columns = df.select_dtypes(include='object').columns.tolist()
print("columns with Object :",object_columns)
```

```
columns with Object : ['STATE', 'PRIVATELABELTENDERFLAG', 'MOSAIC']
```

In [17]:
```
#Checking the shape of dataframe
print("Shape before removing: ", df.shape)
```

```
Shape before removing:  (10000, 117)
```

In [18]:
```
dup_cols = set()

for x in range(df.shape[1]):
```

```
    base_col  = df.iloc[:,x]
    for y in range(x+1,df.shape[1]):
        comp_col = df.iloc[:,y]

        if base_col.equals(comp_col):
            dup_cols.add(df.columns.values[y])
print("Columns with duplicate values: ",dup_cols)
```

```
Columns with duplicate values:  {'7-Cost Only Spend', 'CHILDREN', 'CAPE: HHSIZE: HH: AVERAGE HOUSEHOLD SIZE'}
```

In [19]:
```
df['CHILDREN'].duplicated()
```

Out[19]:
```
0       False
1        True
2        True
3        True
4        True
        ...
9995     True
9996     True
9997     True
9998     True
9999     True
Name: CHILDREN, Length: 10000, dtype: bool
```

In [20]:
```
df['7-Cost Only Spend'].duplicated().sum()
```

Out[20]:
```
9999
```

In [21]:
```
df['CHILDREN'].duplicated().sum()
```

Out[21]:
```
9999
```

In [22]:
```
df['CAPE: HHSIZE: HH: AVERAGE HOUSEHOLD SIZE'].duplicated().sum()
```

Out[22]:
```
9687
```

In [23]:
```
df2 = df.copy()
```

In [24]:
```
#After Removing duplicate columns
df=df.drop(columns = ['7-Cost Only Spend', 'CHILDREN', 'CAPE: HHSIZE: HH: AVERAGE HOUSEHOLD SIZE'])
print(df.head())
```

```
   CUSTOMERID STATE  LCPCOUNT PRIVATELABELTENDERFLAG  TENURE_IN_MONTHS  \
0        5001    TX         1                      N              -9.0
1        5002    OH         0                      Y               9.0
2        5003    TX         0                      N              12.0
3        5004    TN         0                      N              -1.0
4        5005    TX         0                      N              16.0

   CLOSESTSTOREDISTANCE  FEMALE  AGE  HS_DIPLOMA  SOME_COLLEGE  ...  \
0                   NaN       0  NaN           0             0  ...
1              8.728943       0  NaN           0             0  ...
2                   NaN       0  NaN           0             0  ...
3                   NaN       0  NaN           0             0  ...
4                   NaN       0  NaN           0             0  ...

   MDAYREV_L3Y  MDAYREV_L4Y  MDAYQTY_L1Y  MDAYQTY_L2Y  MDAYQTY_L3Y  \
0          0.0          0.0            0            0            0
1          0.0          0.0            0            0            0
2          0.0          0.0            0            0            0
3          0.0          0.0            0            0            0
4          0.0          0.0            0            0            0

   MDAYQTY_L4Y  MDAYSHOPPER_L1Y  MDAYSHOPPER_L2Y  MDAYSHOPPER_L3Y  \
0            0                0                0                0
1            0                0                0                0
2            0                0                0                0
3            0                0                0                0
4            0                0                0                0

   MDAYSHOPPER_L4Y
0                0
1                0
2                0
3                0
4                0

[5 rows x 114 columns]
```

In [25]:
```
#CHeacking the shape after removing the duplicate Columns
print("Shape after removing: ", df.shape)
```

```
Shape after removing:  (10000, 114)
```

In [26]:
```
#Before Removing duplicate columns
print(df2.head())
```

```
   CUSTOMERID STATE  LCPCOUNT PRIVATELABELTENDERFLAG  TENURE_IN_MONTHS  \
0        5001    TX         1                      N              -9.0
1        5002    OH         0                      Y               9.0
2        5003    TX         0                      N              12.0
3        5004    TN         0                      N              -1.0
4        5005    TX         0                      N              16.0

   CLOSESTSTOREDISTANCE  FEMALE  AGE  HS_DIPLOMA  SOME_COLLEGE  ...  \
0                   NaN       0  NaN           0             0  ...
1              8.728943       0  NaN           0             0  ...
2                   NaN       0  NaN           0             0  ...
3                   NaN       0  NaN           0             0  ...
4                   NaN       0  NaN           0             0  ...

   MDAYREV_L3Y  MDAYREV_L4Y  MDAYQTY_L1Y  MDAYQTY_L2Y  MDAYQTY_L3Y  \
0          0.0          0.0            0            0            0
1          0.0          0.0            0            0            0
2          0.0          0.0            0            0            0
3          0.0          0.0            0            0            0
4          0.0          0.0            0            0            0

   MDAYQTY_L4Y  MDAYSHOPPER_L1Y  MDAYSHOPPER_L2Y  MDAYSHOPPER_L3Y  \
0            0                0                0                0
1            0                0                0                0
2            0                0                0                0
3            0                0                0                0
4            0                0                0                0

   MDAYSHOPPER_L4Y
0                0
1                0
2                0
3                0
4                0

[5 rows x 117 columns]
```

In [57]:

```
#Find the columns from the data with constant values
constant_cols=[col for col in df.columns if df[col].nunique()==1]
print("Constant column: ",constant_cols)
```

```
Constant column:  ['MARRIED']
```

In [28]:
```
#removing the constant columns
df_cleaned = df.drop(columns = constant_cols)
#printing the orginal shape dataframe before removing of the constant columns
print("Orginal df Shape:",df.shape)
```

```
Orginal df Shape: (10000, 114)
```

In [29]:
```
#printing the orginal dataframe before removing of the constant columns
print(df.head())
```

```
   CUSTOMERID STATE  LCPCOUNT PRIVATELABELTENDERFLAG  TENURE_IN_MONTHS  \
0        5001    TX         1                      N              -9.0
1        5002    OH         0                      Y               9.0
2        5003    TX         0                      N              12.0
3        5004    TN         0                      N              -1.0
4        5005    TX         0                      N              16.0

   CLOSESTSTOREDISTANCE  FEMALE  AGE  HS_DIPLOMA  SOME_COLLEGE  ...  \
0                   NaN       0  NaN           0             0  ...
1              8.728943       0  NaN           0             0  ...
2                   NaN       0  NaN           0             0  ...
3                   NaN       0  NaN           0             0  ...
4                   NaN       0  NaN           0             0  ...

   MDAYREV_L3Y  MDAYREV_L4Y  MDAYQTY_L1Y  MDAYQTY_L2Y  MDAYQTY_L3Y  \
0          0.0          0.0            0            0            0
1          0.0          0.0            0            0            0
2          0.0          0.0            0            0            0
3          0.0          0.0            0            0            0
4          0.0          0.0            0            0            0

   MDAYQTY_L4Y  MDAYSHOPPER_L1Y  MDAYSHOPPER_L2Y  MDAYSHOPPER_L3Y  \
0            0                0                0                0
1            0                0                0                0
2            0                0                0                0
3            0                0                0                0
4            0                0                0                0

   MDAYSHOPPER_L4Y
0                0
1                0
2                0
3                0
4                0

[5 rows x 114 columns]
```

In [30]:
```
#printing the cleaned dataframe shape after removing of the constant columns
print("Cleaned df Shape:",df_cleaned.shape)
```

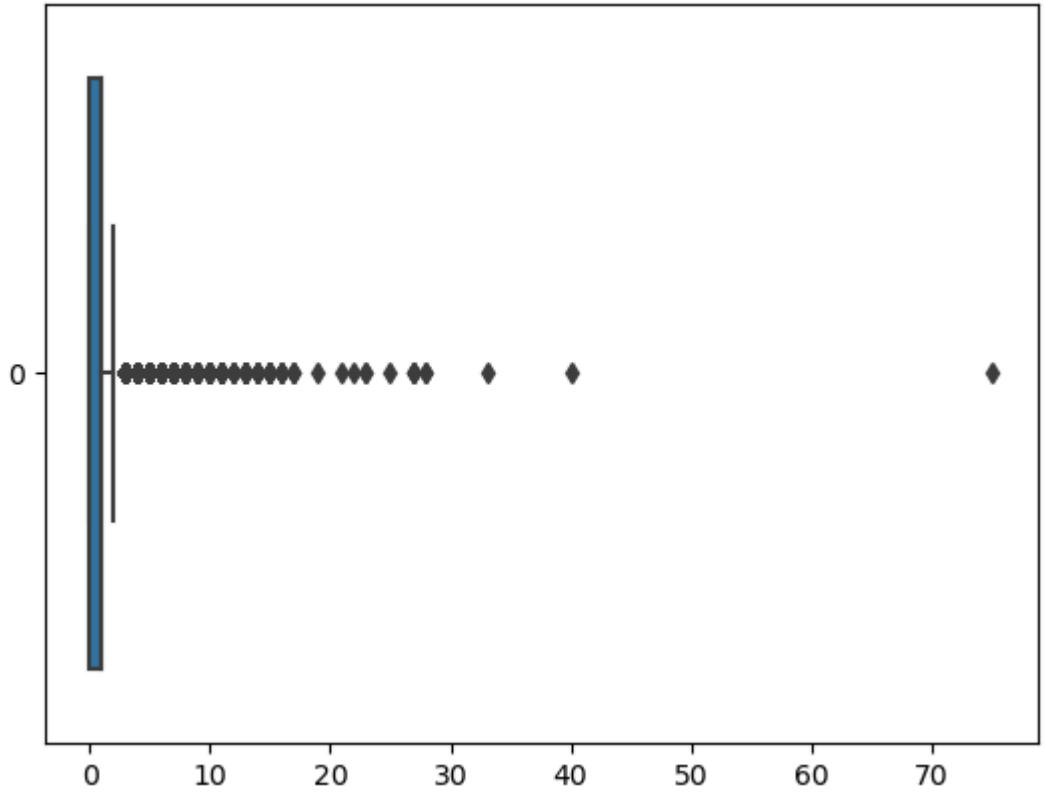
```
Cleaned df Shape: (10000, 113)
```

In [31]:
```
#printing the cleaned dataframe after removing of the constant columns
print(df_cleaned.head())
```

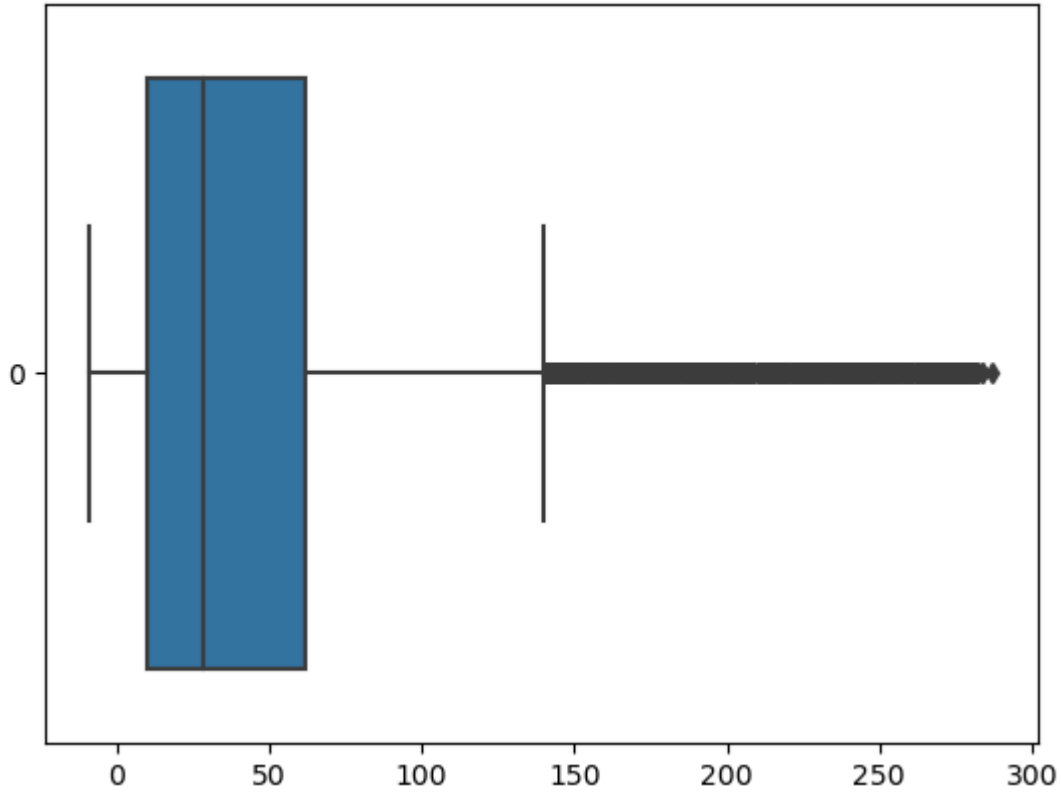
```
   CUSTOMERID STATE  LCPCOUNT PRIVATELABELTENDERFLAG  TENURE_IN_MONTHS  \
0        5001    TX         1                      N              -9.0
1        5002    OH         0                      Y               9.0
2        5003    TX         0                      N              12.0
3        5004    TN         0                      N              -1.0
4        5005    TX         0                      N              16.0

   CLOSESTSTOREDISTANCE  FEMALE  AGE  HS_DIPLOMA  SOME_COLLEGE  ...  \
0                   NaN       0  NaN           0             0  ...
1              8.728943       0  NaN           0             0  ...
2                   NaN       0  NaN           0             0  ...
3                   NaN       0  NaN           0             0  ...
4                   NaN       0  NaN           0             0  ...

   MDAYREV_L3Y  MDAYREV_L4Y  MDAYQTY_L1Y  MDAYQTY_L2Y  MDAYQTY_L3Y  \
0          0.0          0.0            0            0            0
1          0.0          0.0            0            0            0
2          0.0          0.0            0            0            0
3          0.0          0.0            0            0            0
4          0.0          0.0            0            0            0

   MDAYQTY_L4Y  MDAYSHOPPER_L1Y  MDAYSHOPPER_L2Y  MDAYSHOPPER_L3Y  \
0            0                0                0                0
1            0                0                0                0
2            0                0                0                0
3            0                0                0                0
4            0                0                0                0

   MDAYSHOPPER_L4Y
0                0
1                0
2                0
3                0
4                0

[5 rows x 113 columns]
```

In [58]:

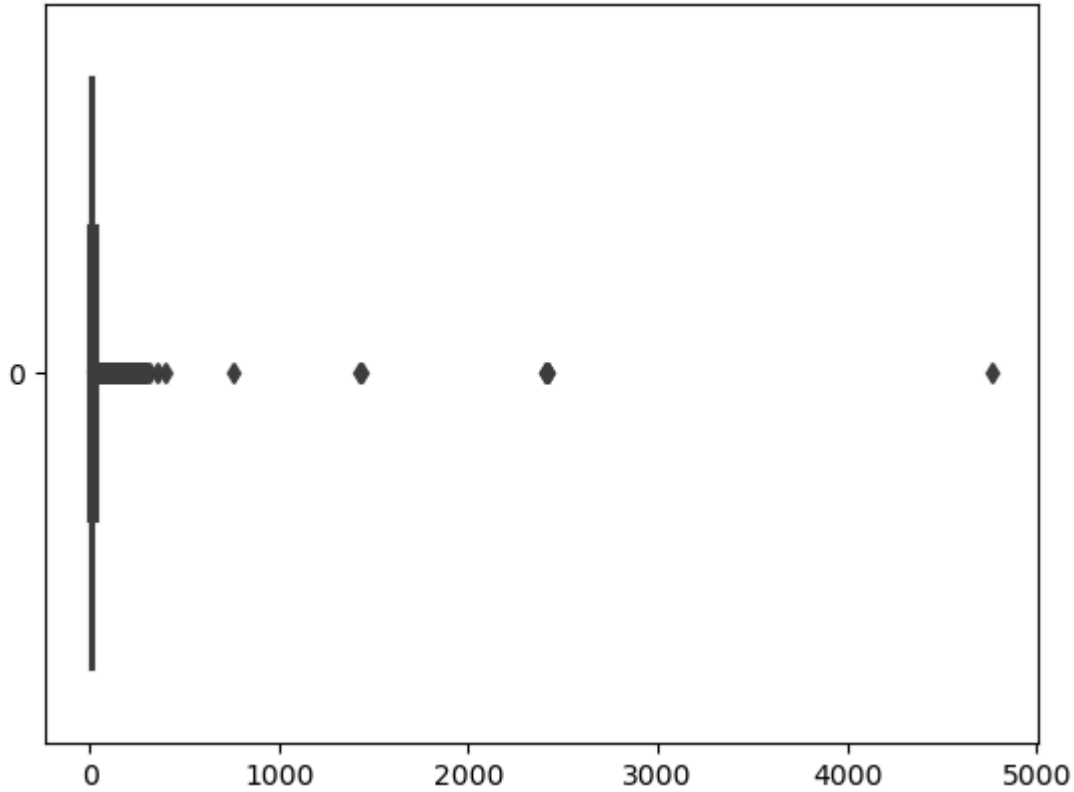
```
#Working on visualizing oulier of all the numric data
import seaborn as sns
#Visualizing LCPCOUNT outliers
box1=sns.boxplot(df_cleaned['LCPCOUNT'],orient='h')
fig = box1.get_figure()
fig.savefig("box1.png",dpi=600)
```



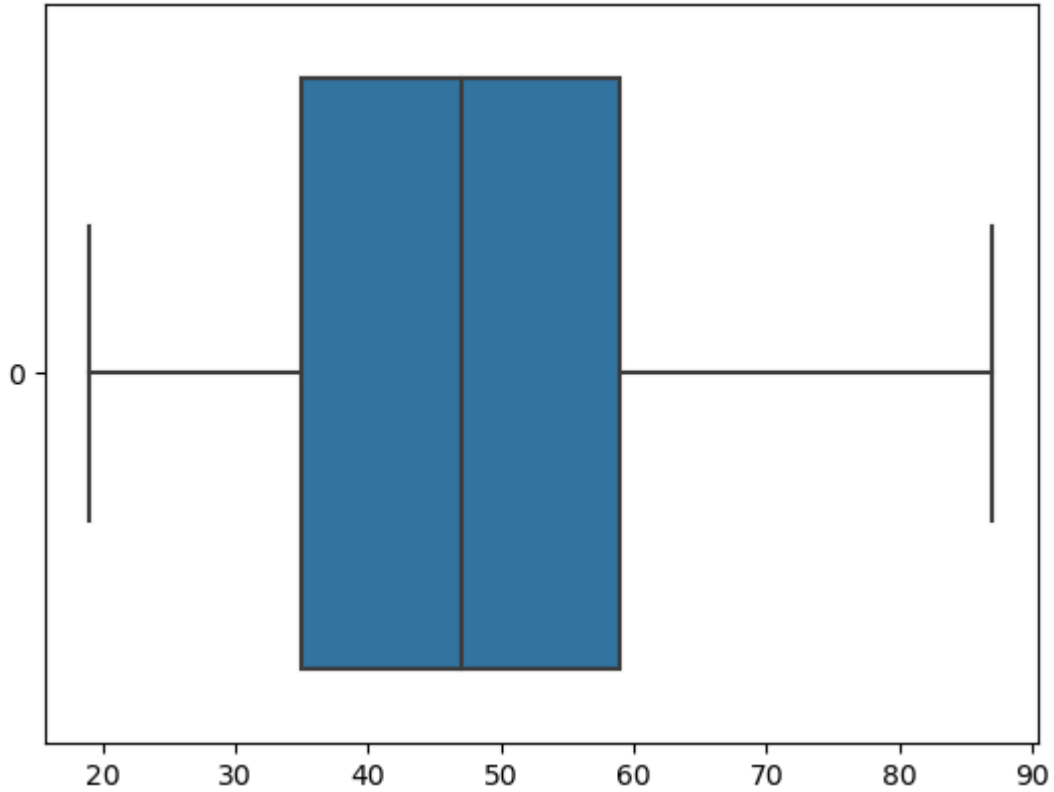
In [59]:
```
#Visualizing TENURE_IN_MONTHS outliers
box2=sns.boxplot(df_cleaned['TENURE_IN_MONTHS'],orient='h')
fig2 = box2.get_figure()
fig2.savefig("box2.png",dpi=600)
```



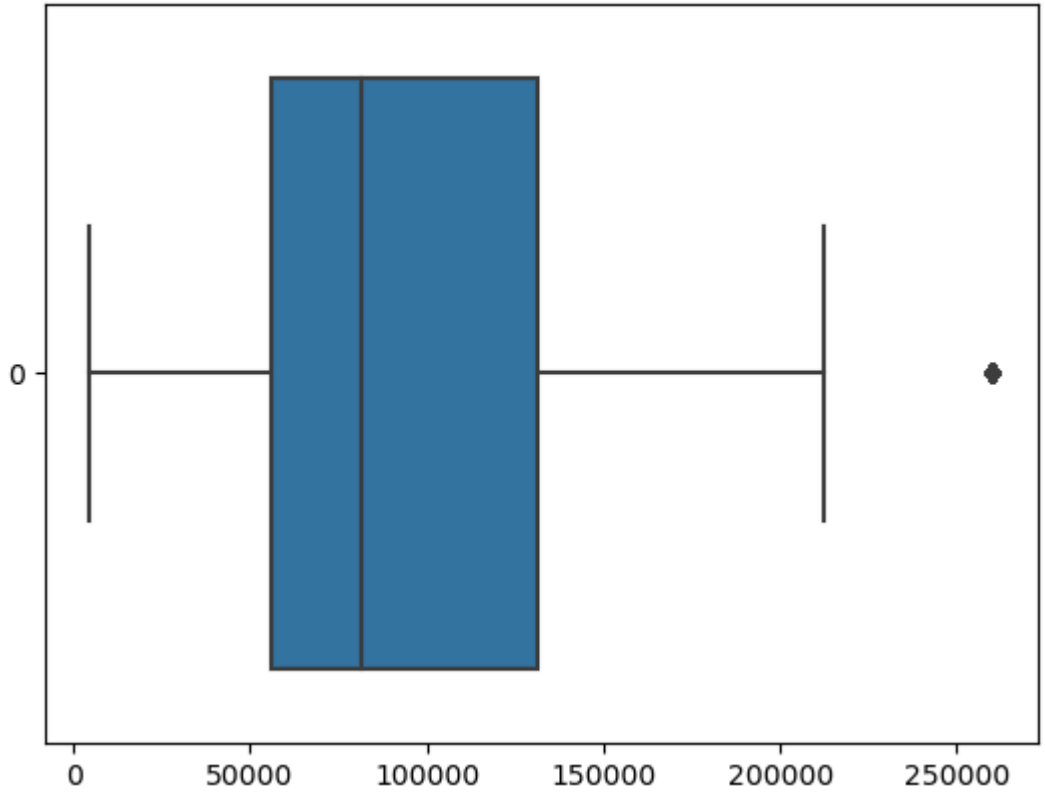
In [60]:
```
#Visualizing CLOSESTSTOREDISTANCE outliers
box3=sns.boxplot(df_cleaned['CLOSESTSTOREDISTANCE'],orient='h')
fig3 = box3.get_figure()
fig3.savefig("box3.png",dpi=600)
```



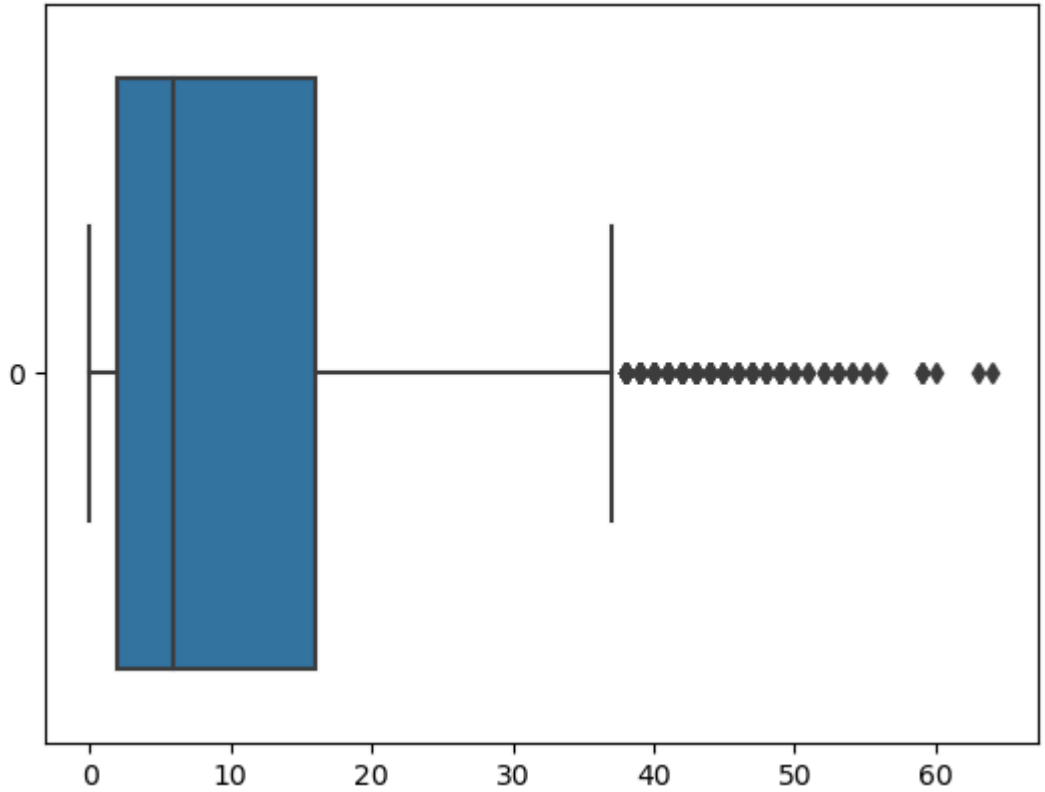
In [61]:
```
#Visualizing AGE outliers
box4=sns.boxplot(df_cleaned['AGE'],orient='h')
fig4 = box4.get_figure()
fig4.savefig("box4.png",dpi=600)
```



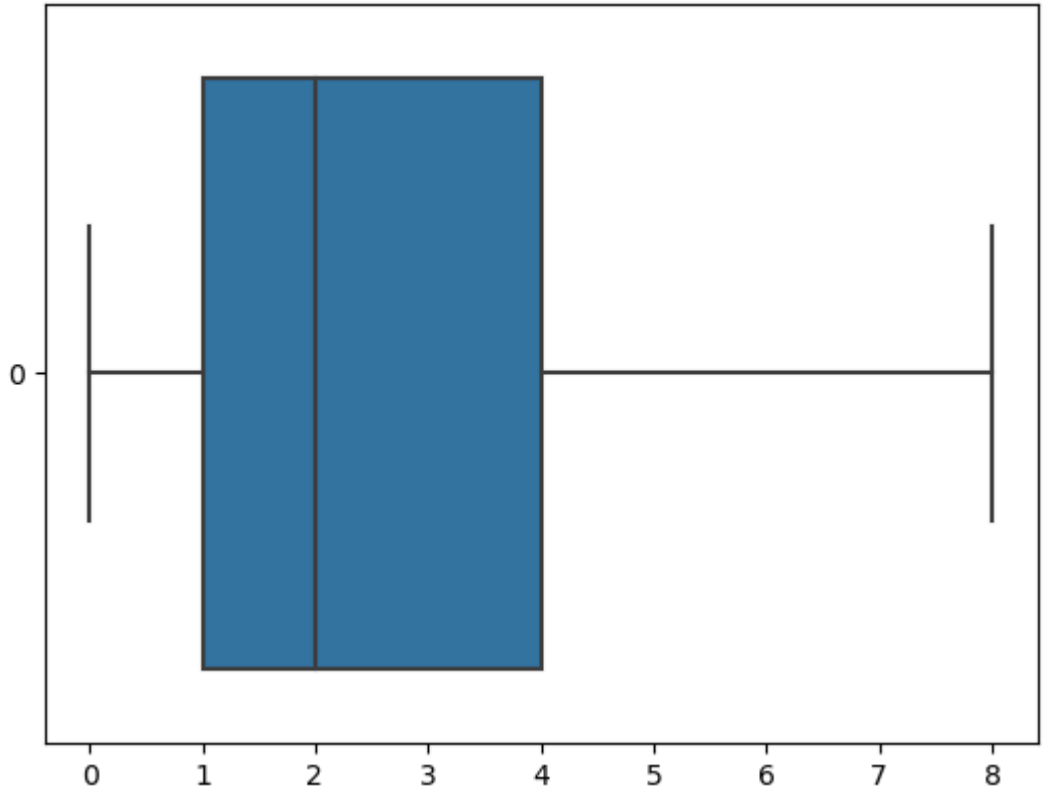
In [62]:
```
#Visualizing INCOME outliers
box5=sns.boxplot(df_cleaned['INCOME'],orient='h')
fig5 = box5.get_figure()
fig5.savefig("box5.png",dpi=600)
```



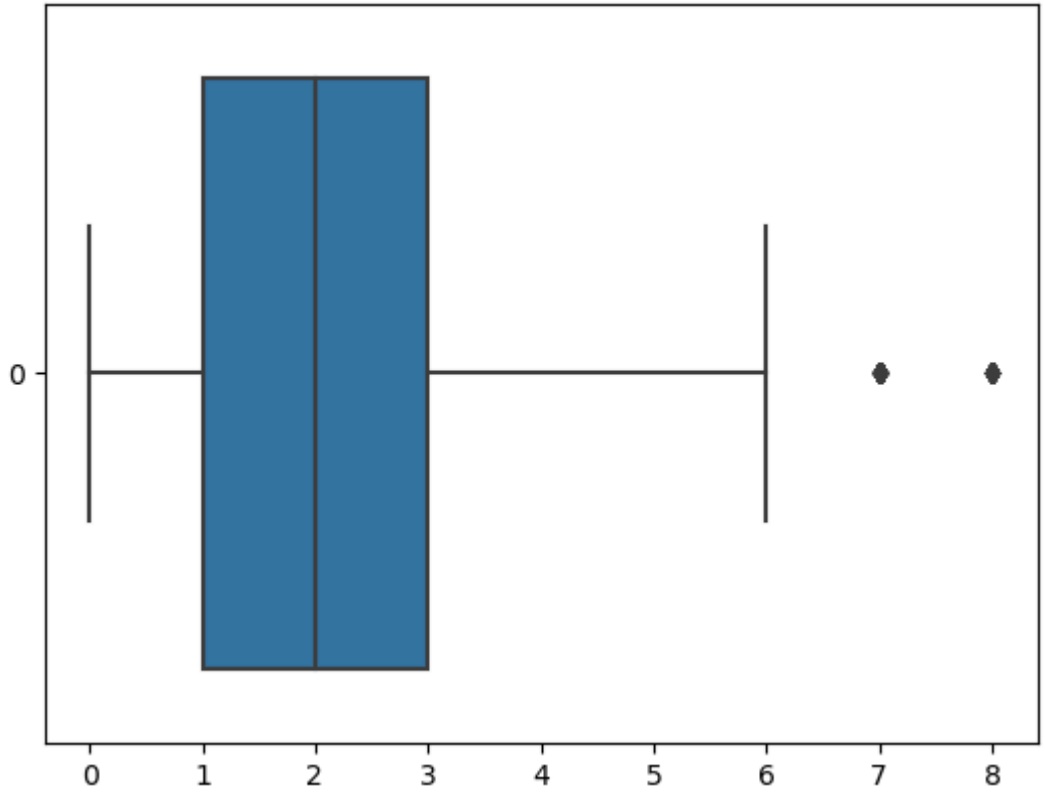
In [63]:
```
#Visualizing LENGTH OF RESIDENCE outliers
box6=sns.boxplot(df_cleaned['LENGTH OF RESIDENCE'],orient='h')
fig6 = box6.get_figure()
fig6.savefig("box6.png",dpi=600)
```



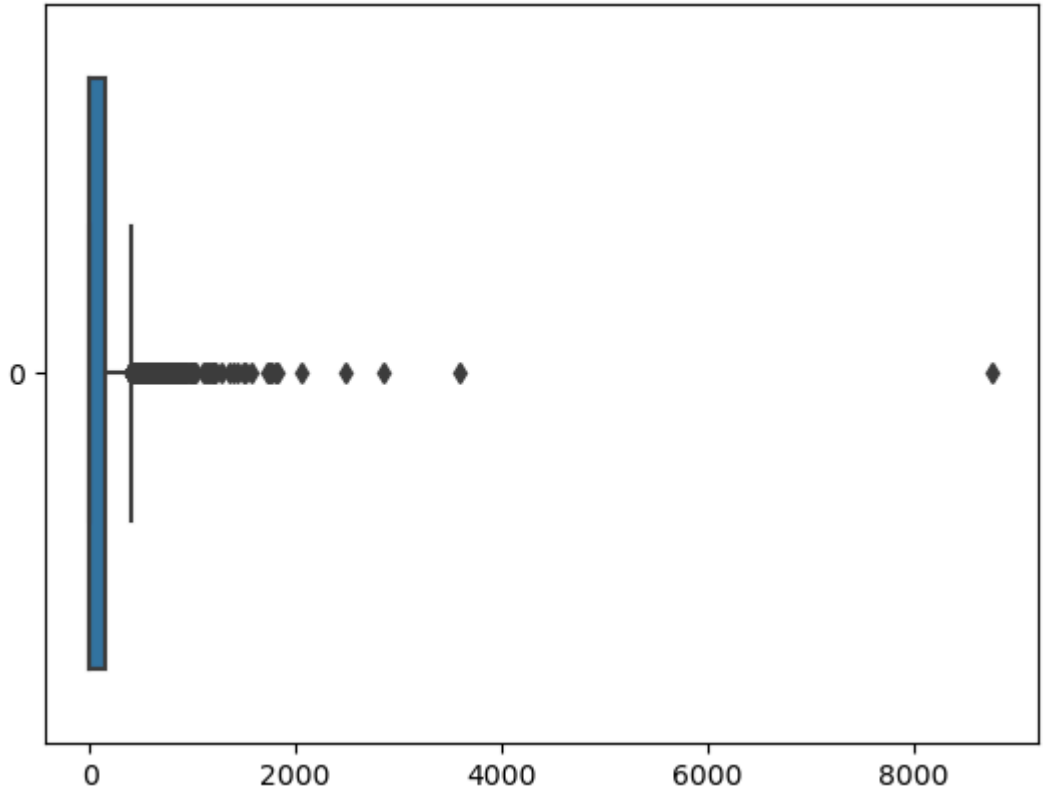
In [64]:
```
#Visualizing NUMBER OF PERSONS IN LIVING UNIT outliers
box7=sns.boxplot(df_cleaned['NUMBER OF PERSONS IN LIVING UNIT'],orient='h')
fig7 = box7.get_figure()
fig7.savefig("box7.png",dpi=600)
```



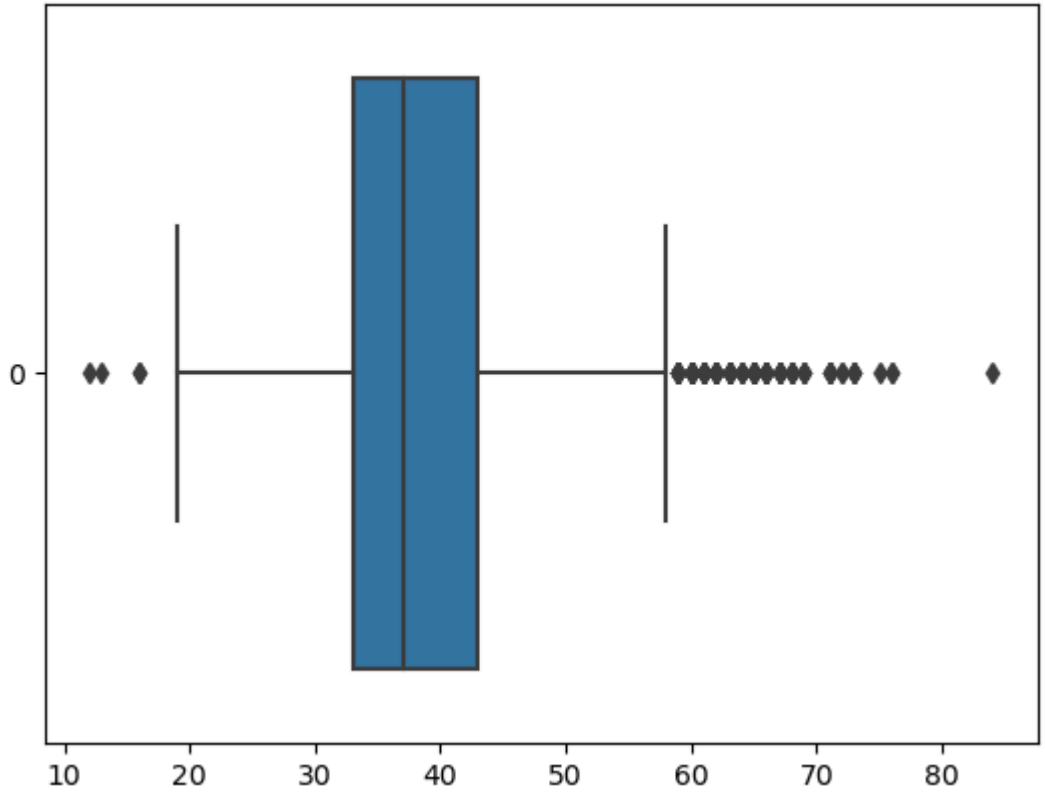
In [65]:
```
#Visualizing NUMBER OF ADULTS IN LIVING UNIT outliers
box8=sns.boxplot(df_cleaned['NUMBER OF ADULTS IN LIVING UNIT'],orient='h')
fig8 = box8.get_figure()
fig8.savefig("box8.png",dpi=600)
```



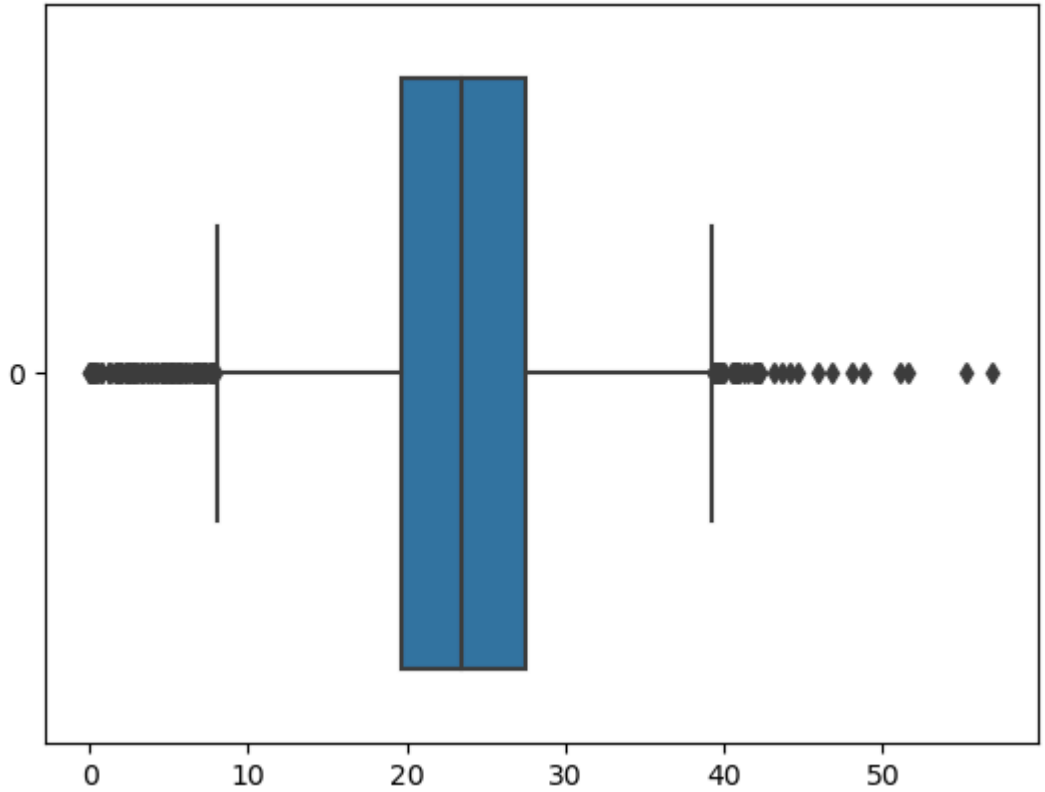
In [66]:
```
#Visualizing MORTGAGE-HOME PURCHASE: HOME PURCHASE PRICE outliers
box9=sns.boxplot(df_cleaned['MORTGAGE-HOME PURCHASE: HOME PURCHASE PRICE'],orient='h')
fig9 = box9.get_figure()
fig9.savefig("box9.png",dpi=600)
```



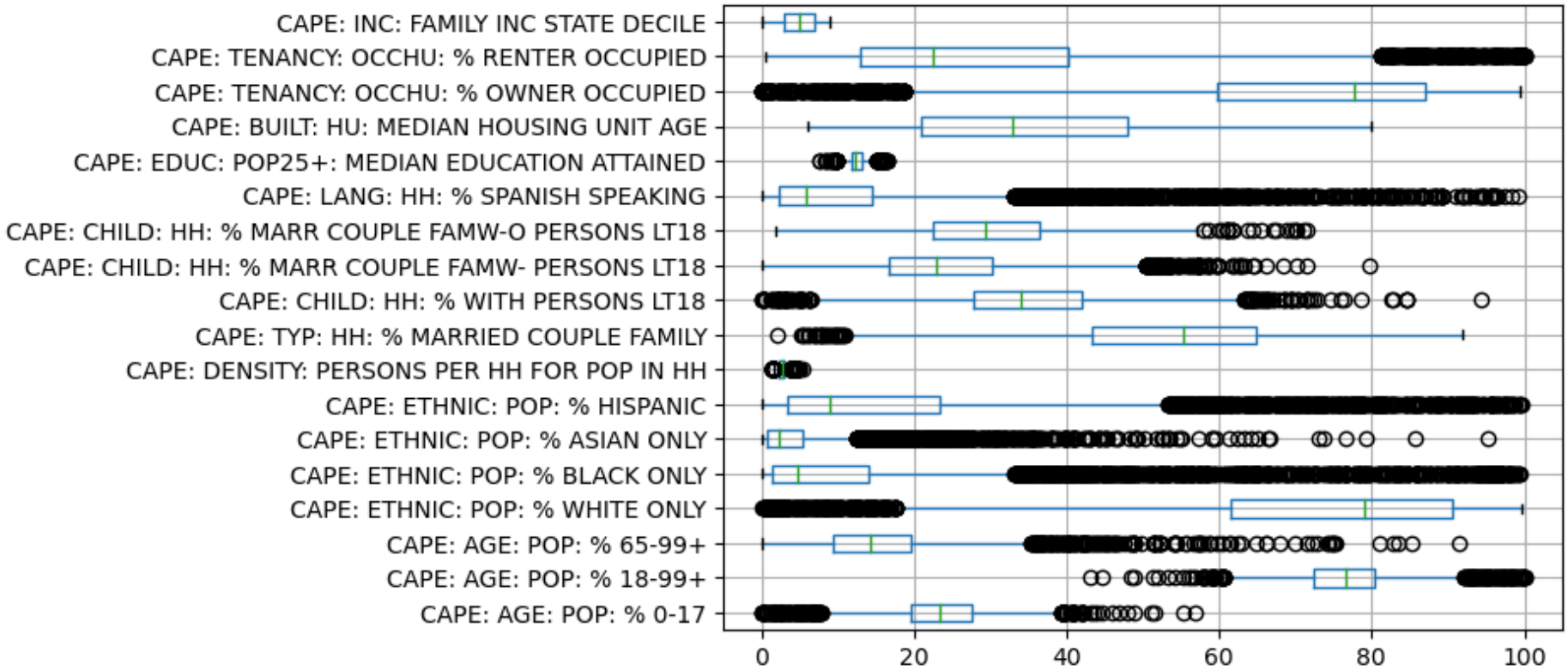
In [67]:
```
#Visualizing CAPE: AGE: POP: MEDIAN AGE outliers
box10=sns.boxplot(df_cleaned['CAPE: AGE: POP: MEDIAN AGE'],orient='h')
fig10 = box10.get_figure()
fig10.savefig("box10.png",dpi=600)
```



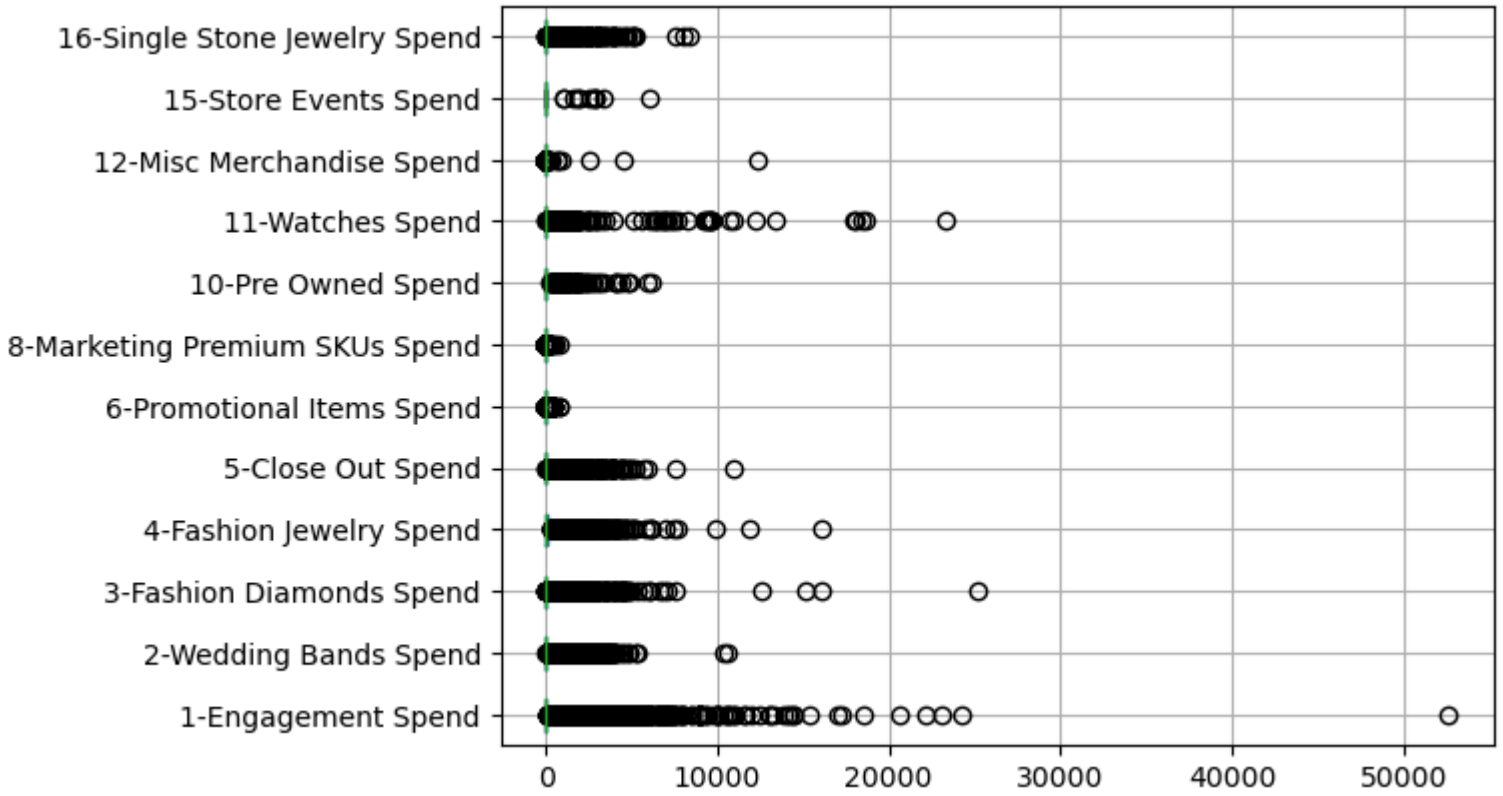
In [68]:
```
#Visualizing CAPE: AGE: POP: % 0-17 outliers
box11=sns.boxplot(df_cleaned['CAPE: AGE: POP: % 0-17'],orient='h')
fig11 = box11.get_figure()
fig11.savefig("box11.png",dpi=600)
```



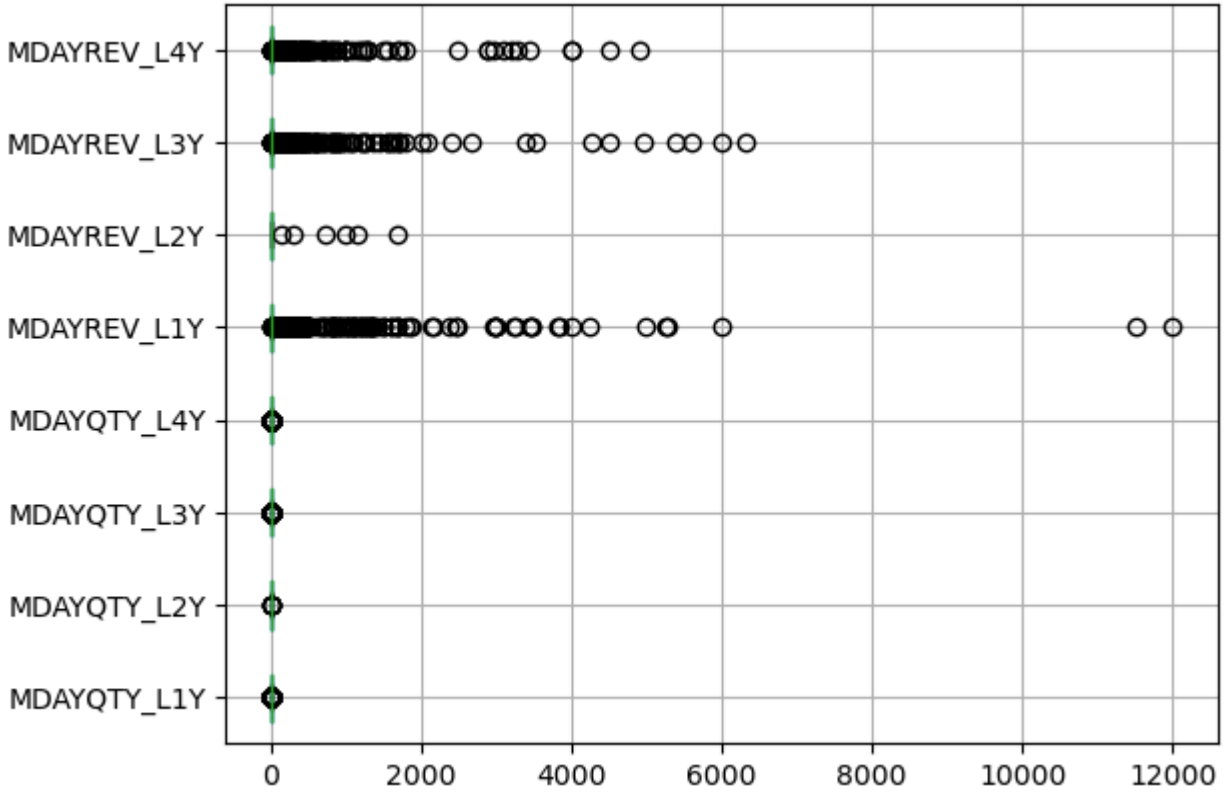
In [76]:
```
data_to_plot= df[['CAPE: AGE: POP: % 0-17', 'CAPE: AGE: POP: % 18-99+',
                  'CAPE: AGE: POP: % 65-99+', 'CAPE: ETHNIC: POP: % WHITE ONLY', 'CAPE: ETHNIC: POP: % BLACK ONLY',
                  'CAPE: ETHNIC: POP: % ASIAN ONLY', 'CAPE: ETHNIC: POP: % HISPANIC',
                  'CAPE: DENSITY: PERSONS PER HH FOR POP IN HH',
                  'CAPE: TYP: HH: % MARRIED COUPLE FAMILY', 'CAPE: CHILD: HH: % WITH PERSONS LT18',
                  'CAPE: CHILD: HH: % MARR COUPLE FAMW- PERSONS LT18', 'CAPE: CHILD: HH: % MARR COUPLE FAMW-O PERSONS LT18',
                  'CAPE: LANG: HH: % SPANISH SPEAKING', 'CAPE: EDUC: POP25+: MEDIAN EDUCATION ATTAINED',
                   'CAPE: BUILT: HU: MEDIAN HOUSING UNIT AGE',
                  'CAPE: TENANCY: OCCHU: % OWNER OCCUPIED', 'CAPE: TENANCY: OCCHU: % RENTER OCCUPIED',
                  'CAPE: INC: FAMILY INC STATE DECILE']]
box12=data_to_plot.boxplot(vert=False)
fig12 = box12.get_figure()
fig12.savefig("box12.png",dpi=2000)
```



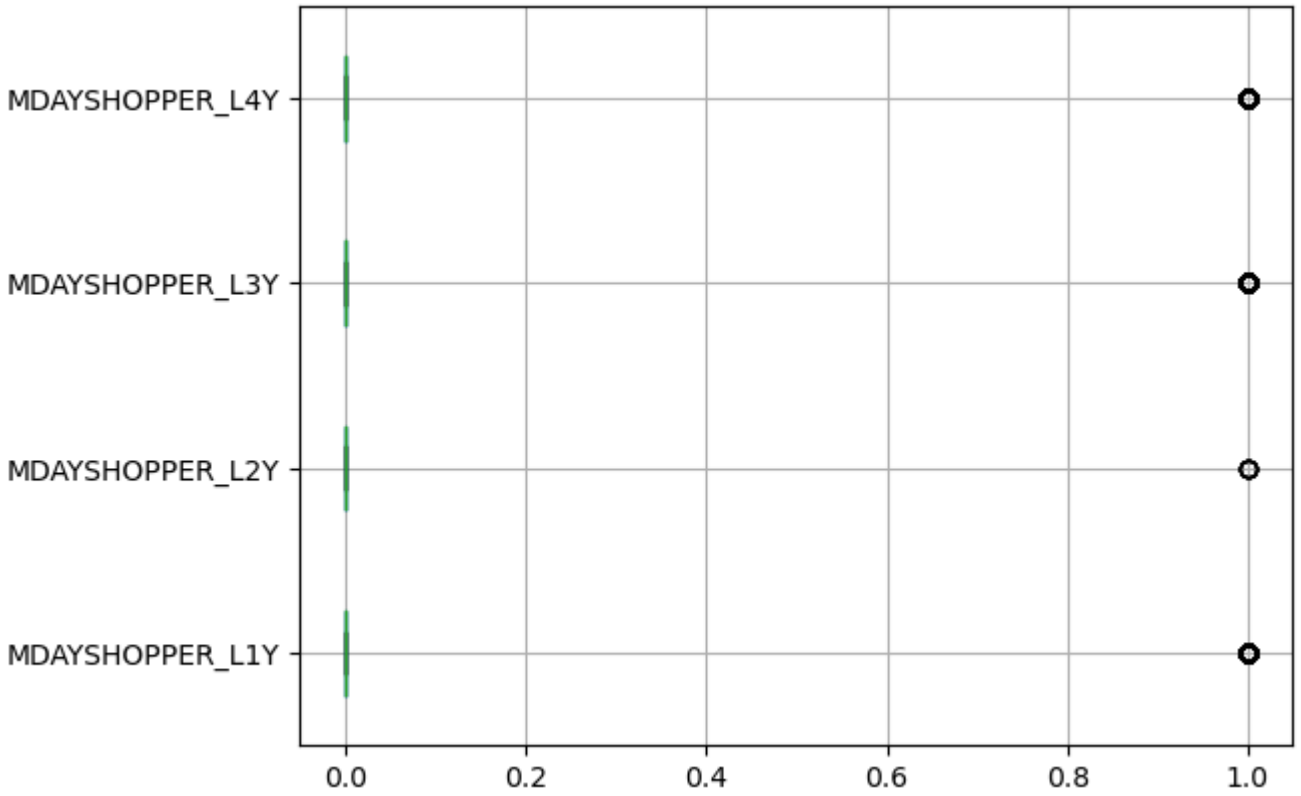
In [70]:
```
data_to_plot2= df[['1-Engagement Spend', '2-Wedding Bands Spend', '3-Fashion Diamonds Spend', '4-Fashion Jewelry Spend',
                   '5-Close Out Spend', '6-Promotional Items Spend', '8-Marketing Premium SKUs Spend',
                   '10-Pre Owned Spend', '11-Watches Spend', '12-Misc Merchandise Spend', '15-Store Events Spend',
                   '16-Single Stone Jewelry Spend'
                ]]
box13=data_to_plot2.boxplot(vert=False)
fig13 = box13.get_figure()
fig13.savefig("box13.png",dpi=600)
```



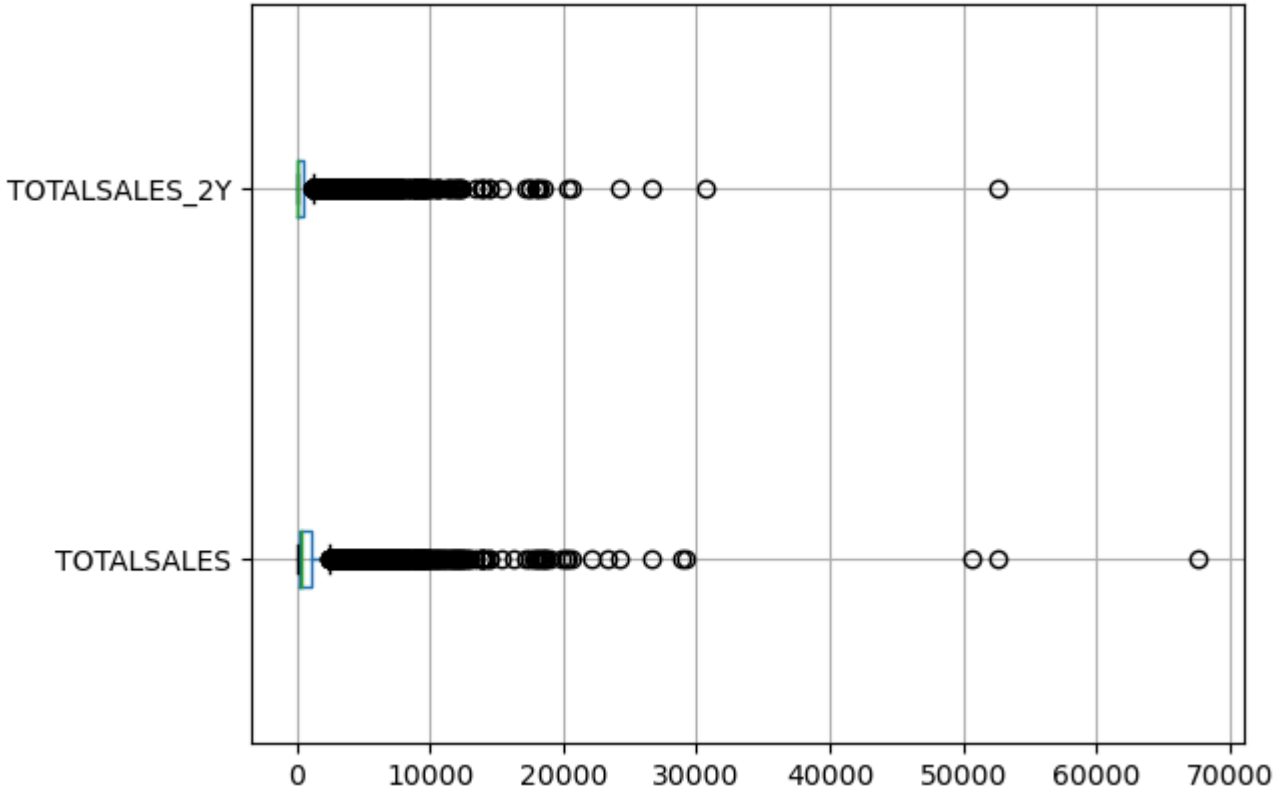
In [71]:
```
data_to_plot3= df[['MDAYQTY_L1Y', 'MDAYQTY_L2Y', 'MDAYQTY_L3Y', 'MDAYQTY_L4Y', 'MDAYREV_L1Y', 'MDAYREV_L2Y', 'MDAYREV_L3Y', 'MDAY
                  ]]
box14=data_to_plot3.boxplot(vert=False)
fig14 = box14.get_figure()
fig14.savefig("box14.png",dpi=600)
```



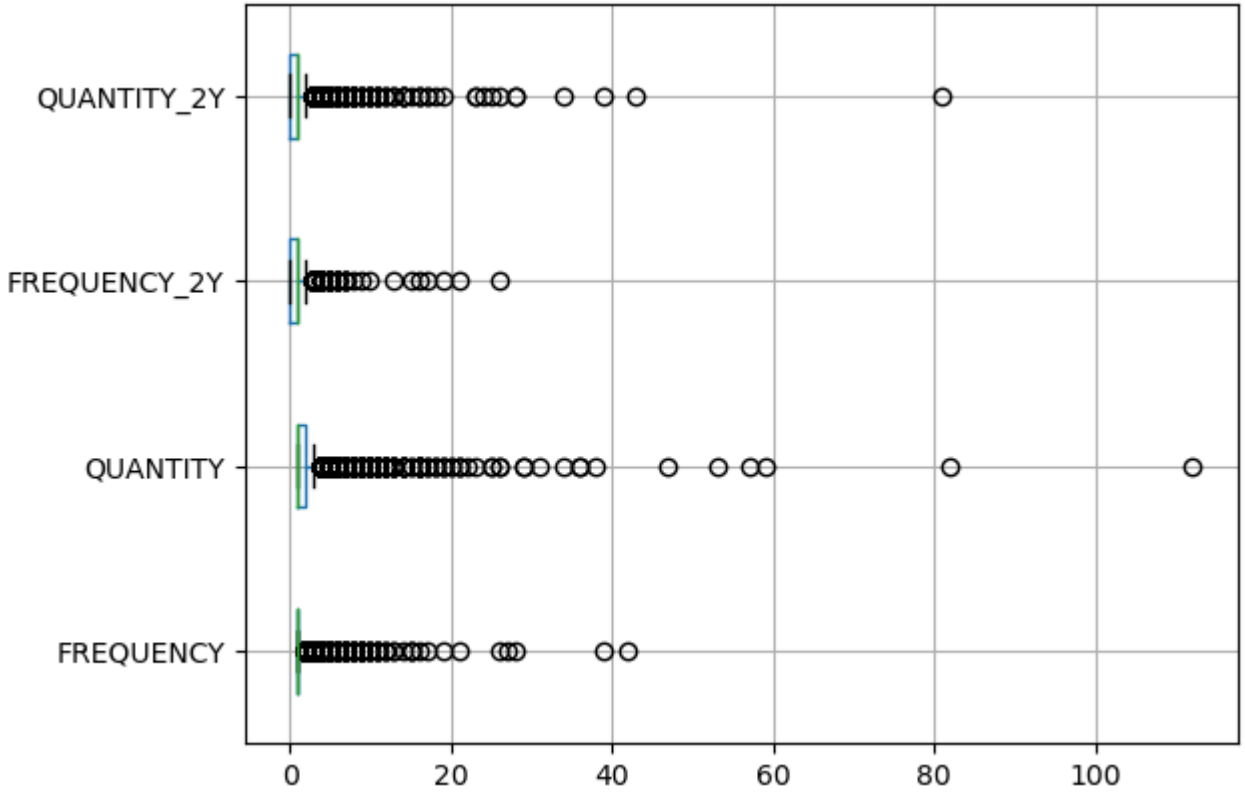
In [72]:
```
data_to_plot4= df[['MDAYSHOPPER_L1Y', 'MDAYSHOPPER_L2Y',
                   'MDAYSHOPPER_L3Y', 'MDAYSHOPPER_L4Y',
                  ]]
box15=data_to_plot4.boxplot(vert=False)
fig15 = box15.get_figure()
fig15.savefig("box15.png",dpi=600)
```



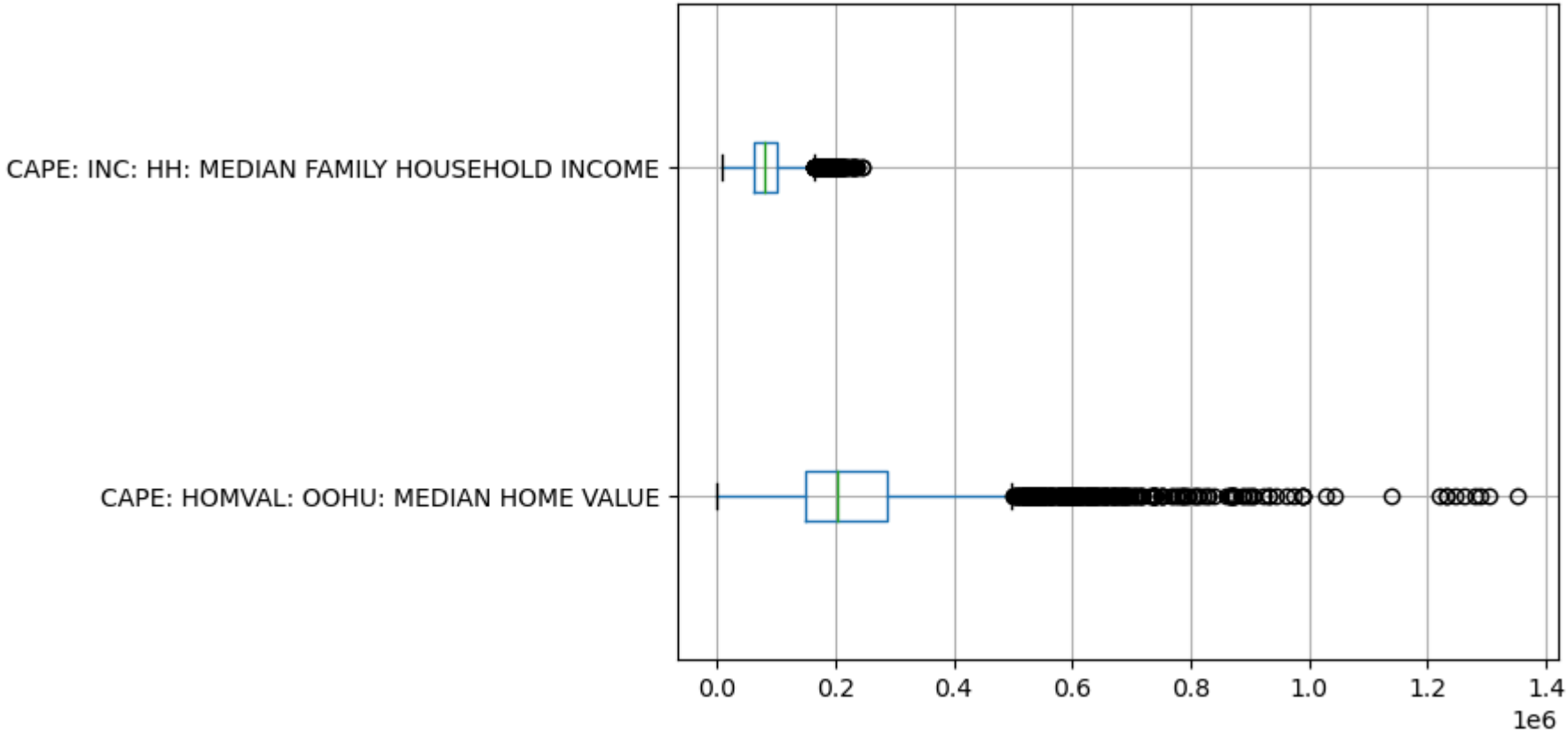
In [73]:
```
data_to_plot5= df[['TOTALSALES','TOTALSALES_2Y']]
box16=data_to_plot5.boxplot(vert=False)
fig16 = box16.get_figure()
fig16.savefig("box16.png",dpi=600)
```



In [74]:
```
data_to_plot6= df[['FREQUENCY','QUANTITY','FREQUENCY_2Y','QUANTITY_2Y']]
box17=data_to_plot6.boxplot(vert=False)
fig17 = box17.get_figure()
fig17.savefig("box17.png",dpi=600)
```



In [75]:
```
data_to_plot7= df[['CAPE: HOMVAL: OOHU: MEDIAN HOME VALUE','CAPE: INC: HH: MEDIAN FAMILY HOUSEHOLD INCOME']]
box18=data_to_plot7.boxplot(vert=False)
fig18 = box18.get_figure()
fig18.savefig("box18.png",dpi=600)
```



In [50]:
```
df2 = df_cleaned.pad(axis = 0)
df_final= df2.fillna(value = df2['AGE'].mean())
df_final= df2.fillna(value = df2['CLOSESTSTOREDISTANCE'].mean())
df_final.head()
```

```
C:\Users\SK\AppData\Local\Temp\ipykernel_7080\803262042.py:1: FutureWarning: DataFrame.pad/Series.pad is deprecated. Use DataFra
me.ffill/Series.ffill instead
  df2 = df_cleaned.pad(axis = 0)
```

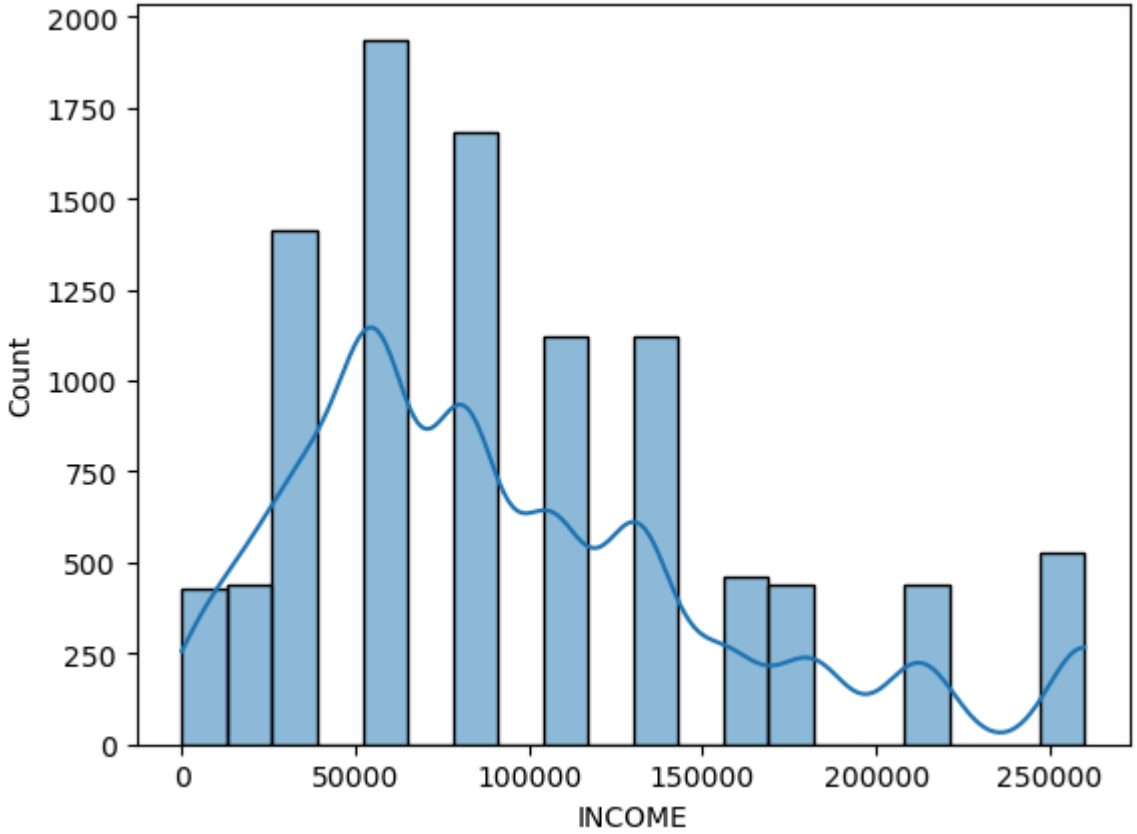
Out[50]:

	CUSTOMERID	STATE	LCPCOUNT	PRIVATELABELTENDERFLAG	TENURE_IN_MONTHS	CLOSESTSTOREDISTANCE	FEMALE	AGE	HS_DIPLOMA	SOM
0	5001	TX	1	N	-9.0	21.855361	0	21.855361	0	
1	5002	OH	0	Y	9.0	8.728943	0	21.855361	0	
2	5003	TX	0	N	12.0	8.728943	0	21.855361	0	
3	5004	TN	0	N	-1.0	8.728943	0	21.855361	0	
4	5005	TX	0	N	16.0	8.728943	0	21.855361	0	

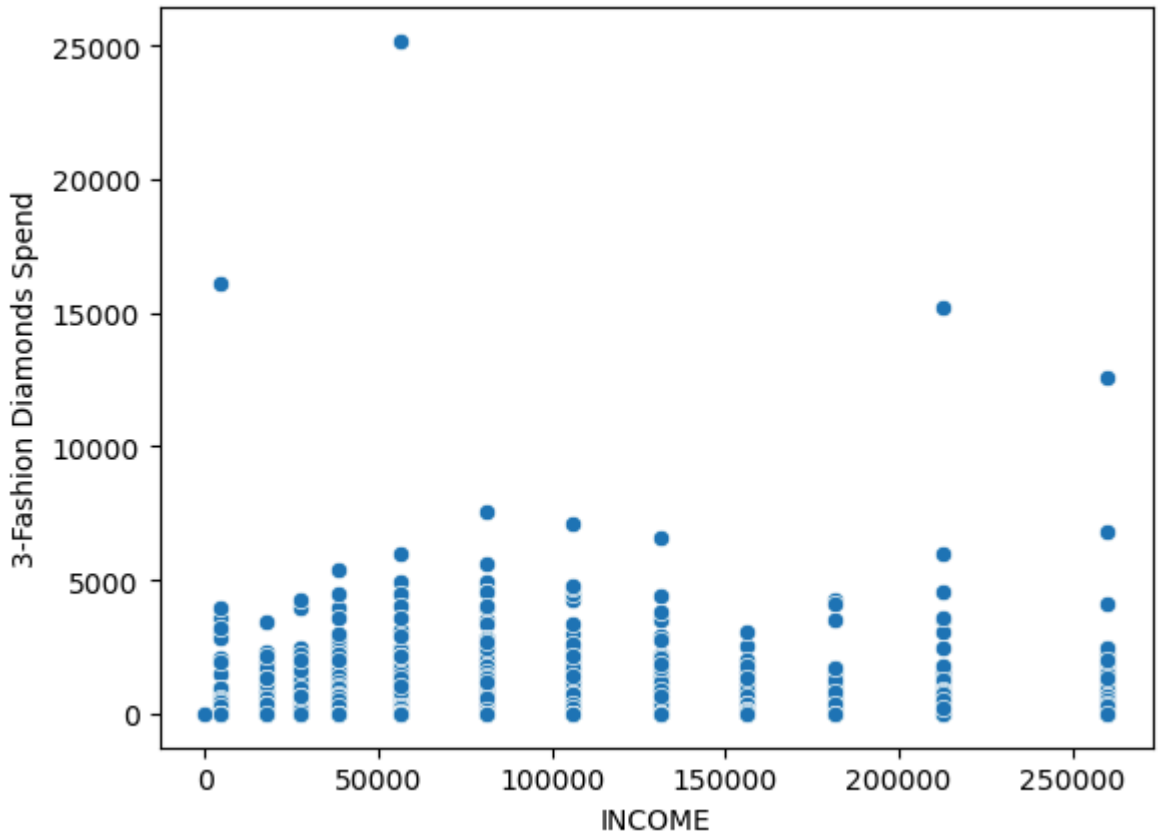
5 rows × 113 columns

In [77]:
```
dist1=sns.histplot(df_final["INCOME"],kde=True ,bins = 20)
fig1 = dist1.get_figure()
fig1.savefig("dist1.png",dpi=600)
```

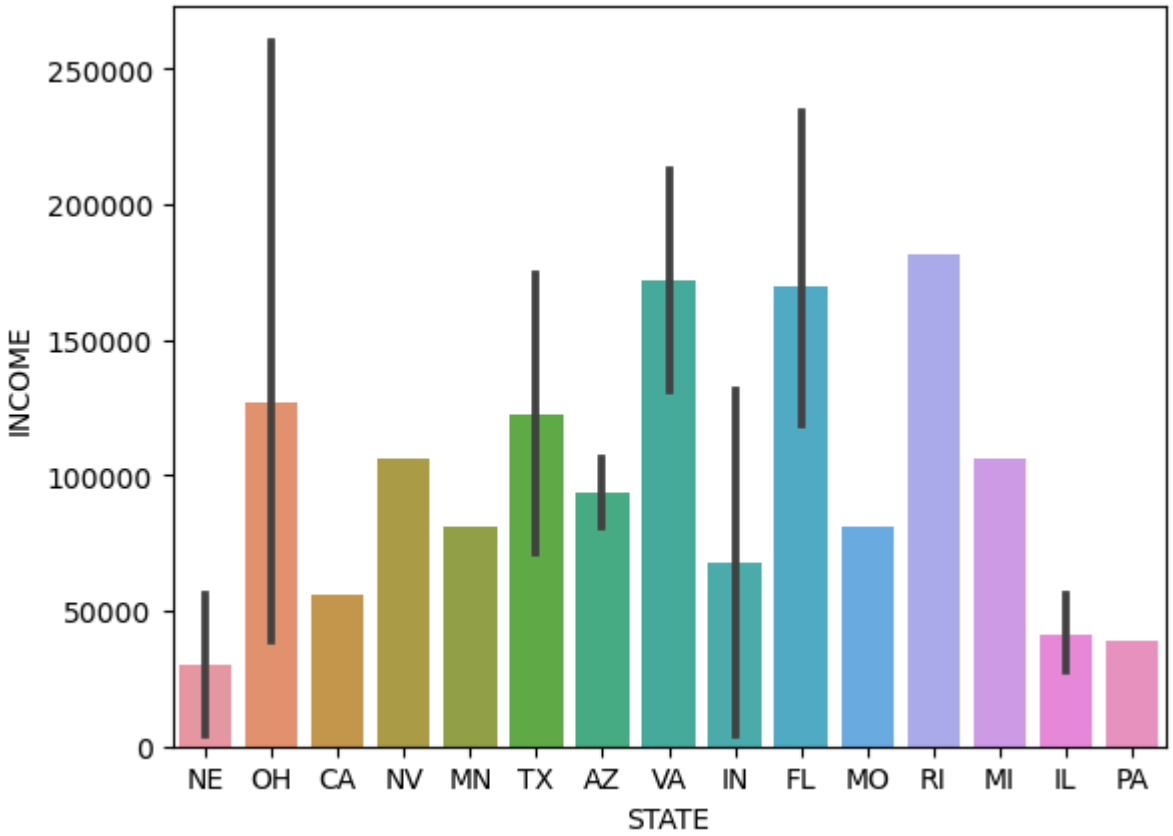
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [78]:
```
dist2=sns.scatterplot(x="INCOME",y="3-Fashion Diamonds Spend",data = df_final)
fig2 = dist2.get_figure()
fig2.savefig("dist2.png",dpi=600)
```



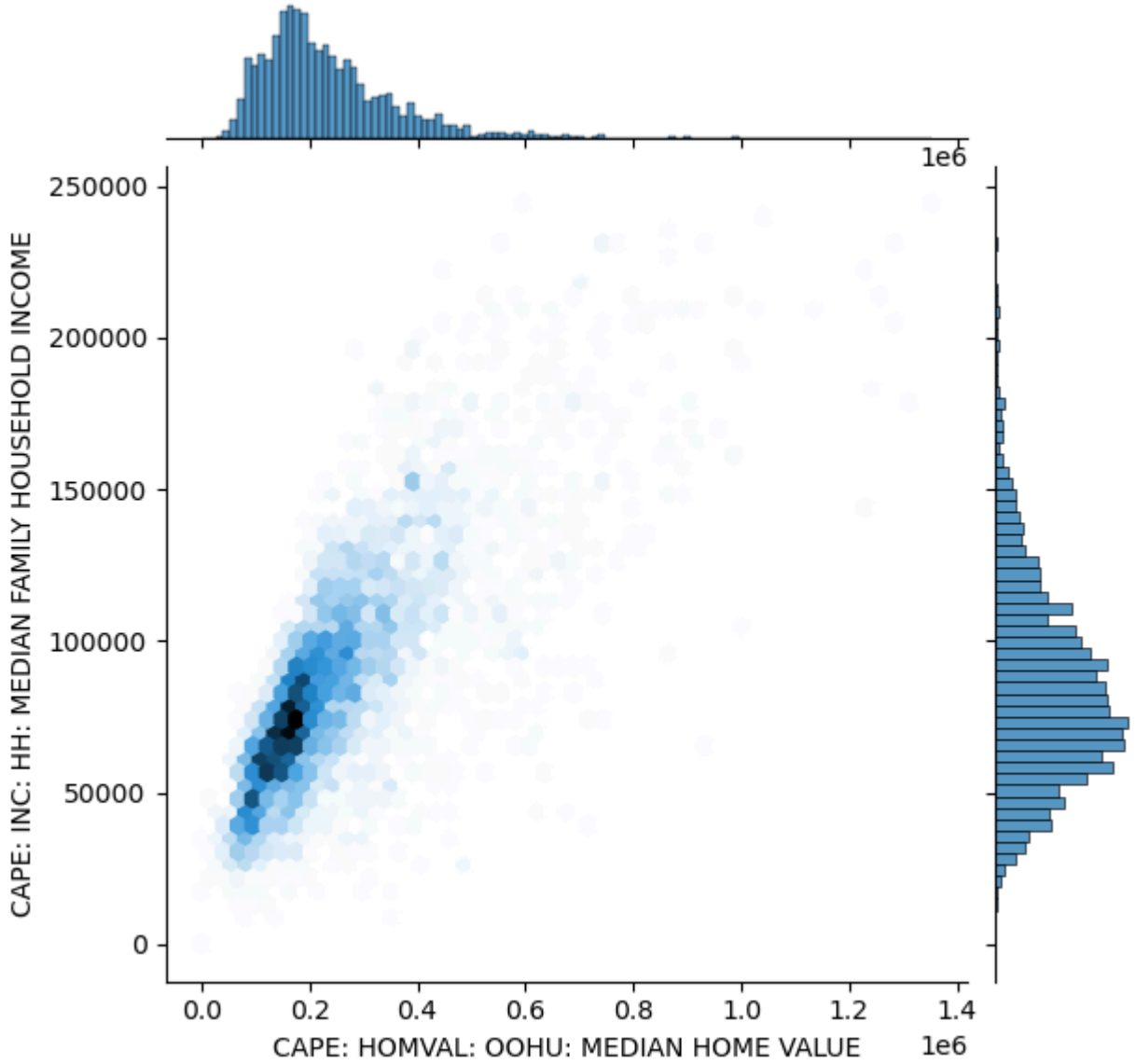
In [79]:
```
dist3=sns.barplot(x="STATE",y="INCOME",data = df_final.nlargest(30,'TOTALSALES'))
fig3 = dist3.get_figure()
fig3.savefig("dist3.png",dpi=600)
```



In [84]:
```
sns.jointplot(x="CAPE: HOMVAL: OOHU: MEDIAN HOME VALUE",y = "CAPE: INC: HH: MEDIAN FAMILY HOUSEHOLD INCOME",
              data = df_final,kind="hex")
```

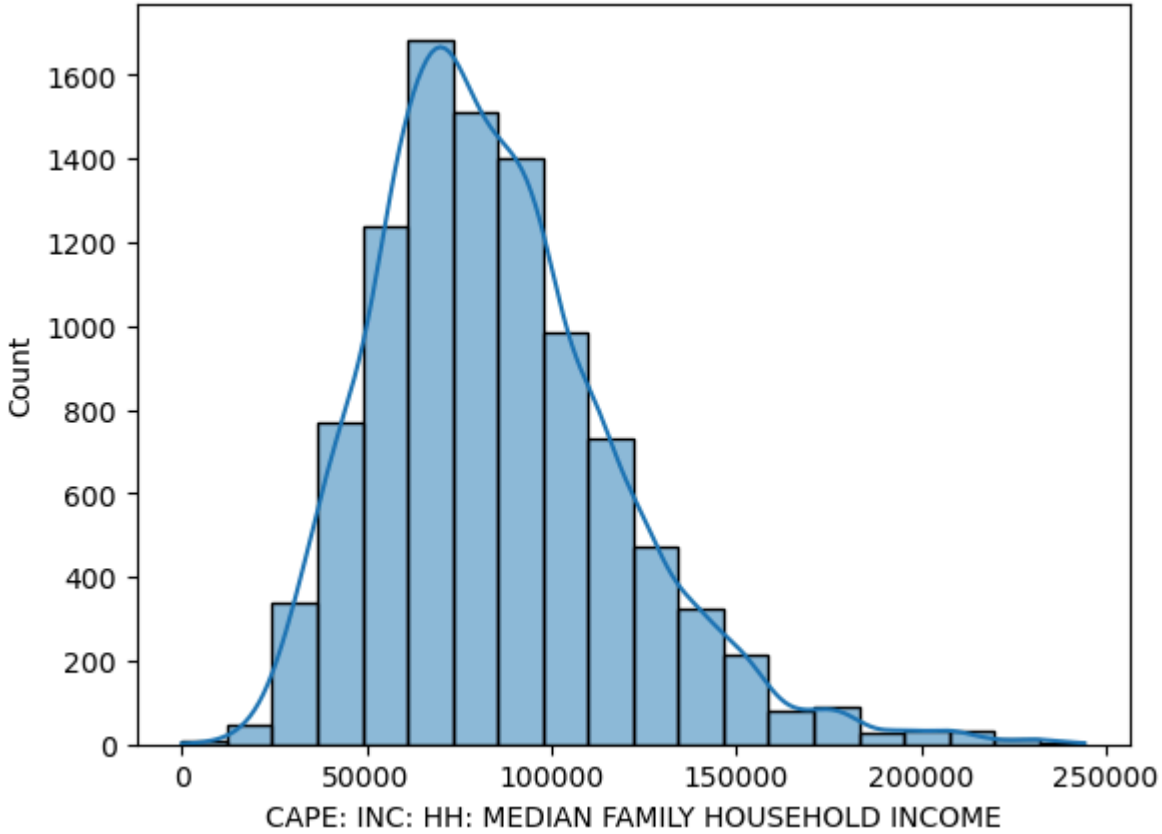
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[84]:
```
<seaborn.axisgrid.JointGrid at 0x2d5519dbfd0>
```



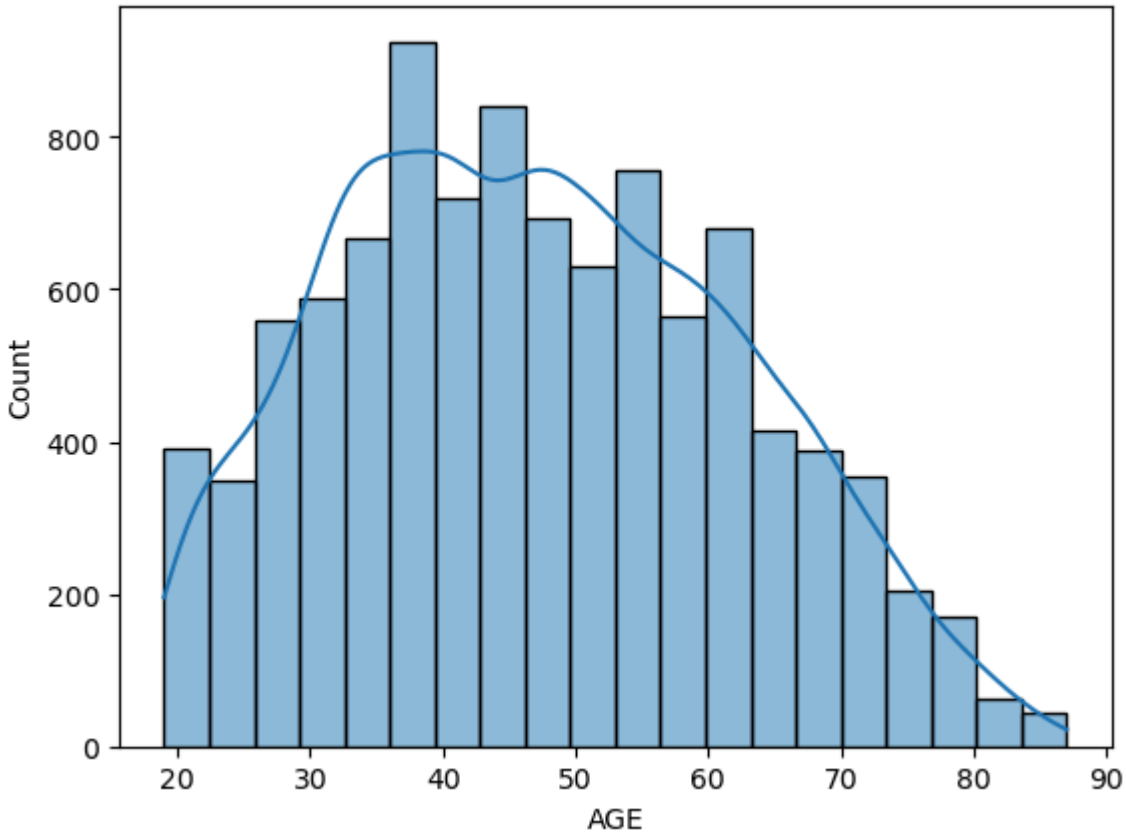
In [82]:
```
dist5=sns.histplot(df_final["CAPE: INC: HH: MEDIAN FAMILY HOUSEHOLD INCOME"],kde=True ,bins = 20)
fig5 = dist5.get_figure()
fig5.savefig("dist5.png",dpi=600)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [83]:
```
dist6=sns.histplot(df_final["AGE"],kde=True ,bins = 20)
fig6 = dist6.get_figure()
fig6.savefig("dist6.png",dpi=600)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



By Shubhendra Kumar

kshubhendra8860@gmail.com

In [ ]:
```
Thank You
```