Reading the xlsx file

In [1]:
```
import pandas as pd
```

In [2]:
```
eda_df=pd.read_excel("./EDA_Dataset.xlsx")
eda_df
```

Out[2]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
0	5524	1957	Graduation	Single	58138.0	0	0	2012
1	2174	1954	Graduation	Single	46344.0	1	1	2014
2	4141	1965	Graduation	Together	71613.0	0	0	2013
3	6182	1984	Graduation	Together	26646.0	1	0	2014
4	5324	1981	PhD	Married	58293.0	1	0	2014
...	...	...	...	...	...	...	...	
2235	10870	1967	Graduation	Married	61223.0	0	1	2013
2236	4001	1946	PhD	Together	64014.0	2	1	2014
2237	7270	1981	Graduation	Divorced	56981.0	0	0	2014
2238	8235	1956	Master	Together	69245.0	0	1	2014
2239	9405	1954	PhD	Married	52869.0	1	1	2012

2240 rows × 29 columns

Creating a duplicate copy of dataset to preserve the original dataset

In [3]:
```
dupl_df=eda_df.copy()
dupl_df
```

Out[3]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
0	5524	1957	Graduation	Single	58138.0	0	0	2012-
1	2174	1954	Graduation	Single	46344.0	1	1	2014-
2	4141	1965	Graduation	Together	71613.0	0	0	2013-
3	6182	1984	Graduation	Together	26646.0	1	0	2014-
4	5324	1981	PhD	Married	58293.0	1	0	2014-
...	...	...	...	...	...	...	...	
2235	10870	1967	Graduation	Married	61223.0	0	1	2013-
2236	4001	1946	PhD	Together	64014.0	2	1	2014-
2237	7270	1981	Graduation	Divorced	56981.0	0	0	2014-
2238	8235	1956	Master	Together	69245.0	0	1	2014-
2239	9405	1954	PhD	Married	52869.0	1	1	2012-

2240 rows × 29 columns

Size of data

In [4]:
```
dupl_df.shape
```

Out[4]:
```
(2240, 29)
```

Disk space

In [5]:
```
dupl_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   ID                   2240 non-null   int64         
 1   Year_Birth           2240 non-null   int64         
 2   Education            2240 non-null   object        
 3   Marital_Status       2240 non-null   object        
 4   Income               2216 non-null   float64       
 5   Kidhome              2240 non-null   int64         
 6   Teenhome             2240 non-null   int64         
 7   Dt_Customer          2240 non-null   datetime64[ns]
 8   Recency              2240 non-null   int64         
 9   MntWines             2240 non-null   int64         
 10  MntFruits            2240 non-null   int64         
 11  MntMeatProducts      2240 non-null   int64         
 12  MntFishProducts      2240 non-null   int64         
 13  MntSweetProducts     2240 non-null   int64         
 14  MntGoldProds         2240 non-null   int64         
 15  NumDealsPurchases    2240 non-null   int64         
 16  NumWebPurchases      2240 non-null   int64         
 17  NumCatalogPurchases  2240 non-null   int64         
 18  NumStorePurchases    2240 non-null   int64         
 19  NumWebVisitsMonth    2240 non-null   int64         
 20  AcceptedCmp3         2240 non-null   int64         
 21  AcceptedCmp4         2240 non-null   int64         
 22  AcceptedCmp5         2240 non-null   int64         
 23  AcceptedCmp1         2240 non-null   int64         
 24  AcceptedCmp2         2240 non-null   int64         
 25  Complain             2240 non-null   int64         
 26  Z_CostContact        2240 non-null   int64         
 27  Z_Revenue            2240 non-null   int64         
 28  Response             2240 non-null   int64         
dtypes: datetime64[ns](1), float64(1), int64(25), object(2)
memory usage: 507.6+ KB
```

Datatype

In [6]:

```
dupl_df.dtypes
```

Out[6]:
```
ID                              int64
Year_Birth                      int64
Education                      object
Marital_Status                 object
Income                        float64
Kidhome                         int64
Teenhome                        int64
Dt_Customer            datetime64[ns]
Recency                         int64
MntWines                        int64
MntFruits                       int64
MntMeatProducts                 int64
MntFishProducts                 int64
MntSweetProducts                int64
MntGoldProds                    int64
NumDealsPurchases               int64
NumWebPurchases                 int64
NumCatalogPurchases             int64
NumStorePurchases               int64
NumWebVisitsMonth               int64
AcceptedCmp3                    int64
AcceptedCmp4                    int64
AcceptedCmp5                    int64
AcceptedCmp1                    int64
AcceptedCmp2                    int64
Complain                        int64
Z_CostContact                   int64
Z_Revenue                       int64
Response                        int64
dtype: object
```

Data is quite big having 29 columns and 2240 rows with datatypes:datetime64ns, float64(1), int64(25), object(2) and consuming 507.6+kb all columns are not null

In [ ]:

Columns

In [7]:
```
dupl_df.columns
dupl_df.nunique()
```

Out[7]:
```
ID                     2240
Year_Birth               59
Education                 5
Marital_Status            8
Income                 1974
Kidhome                   3
Teenhome                  3
Dt_Customer             663
Recency                 100
MntWines                776
MntFruits               158
MntMeatProducts         558
MntFishProducts         182
MntSweetProducts        177
MntGoldProds            213
NumDealsPurchases        15
NumWebPurchases          15
NumCatalogPurchases      14
NumStorePurchases        14
NumWebVisitsMonth        16
AcceptedCmp3              2
AcceptedCmp4              2
AcceptedCmp5              2
AcceptedCmp1              2
AcceptedCmp2              2
Complain                  2
Z_CostContact             1
Z_Revenue                 1
Response                  2
dtype: int64
```

In [8]:
```
dupl_df
```

Out[8]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
0	5524	1957	Graduation	Single	58138.0	0	0	2012-
1	2174	1954	Graduation	Single	46344.0	1	1	2014-
2	4141	1965	Graduation	Together	71613.0	0	0	2013-
3	6182	1984	Graduation	Together	26646.0	1	0	2014-
4	5324	1981	PhD	Married	58293.0	1	0	2014-
...	...	...	...	...	...	...	...	
2235	10870	1967	Graduation	Married	61223.0	0	1	2013-
2236	4001	1946	PhD	Together	64014.0	2	1	2014-
2237	7270	1981	Graduation	Divorced	56981.0	0	0	2014-
2238	8235	1956	Master	Together	69245.0	0	1	2014-
2239	9405	1954	PhD	Married	52869.0	1	1	2012-

2240 rows × 29 columns

Conclusion by reading the dataset and column names

We can assume this dataset as a complete customer expolaratory record of a supermarket having unique customer ids, birth year,education,income etc.

In [9]:
```
dupl_df["Complain"].unique()
```

Out[9]:
```
array([0, 1], dtype=int64)
```

In [10]:
```
#Checking all the unique values of columns
pd.Series({c: dupl_df[c].unique() for c in dupl_df})
```

Out[10]:
```
ID                      [5524, 2174, 4141, 6182, 5324, 7446, 965, 6177...
Year_Birth              [1957, 1954, 1965, 1984, 1981, 1967, 1971, 198...
Education                      [Graduation, PhD, Master, Basic, 2n Cycle]
Marital_Status          [Single, Together, Married, Divorced, Widow, A...
Income                  [58138.0, 46344.0, 71613.0, 26646.0, 58293.0, ...
Kidhome                                                         [0, 1, 2]
Teenhome                                                        [0, 1, 2]
Dt_Customer             [2012-09-04 00:00:00, 2014-03-08 00:00:00, 201...
Recency                 [58, 38, 26, 94, 16, 34, 32, 19, 68, 11, 59, 8...
MntWines                [635, 11, 426, 173, 520, 235, 76, 14, 28, 5, 6...
MntFruits               [88, 1, 49, 4, 43, 42, 65, 10, 0, 5, 16, 61, 2...
MntMeatProducts         [546, 6, 127, 20, 118, 98, 164, 56, 24, 11, 48...
MntFishProducts         [172, 2, 111, 10, 46, 0, 50, 3, 1, 11, 225, 6,...
MntSweetProducts        [88, 1, 21, 3, 27, 42, 49, 2, 112, 5, 68, 13, ...
MntGoldProds            [88, 6, 42, 5, 15, 14, 27, 23, 2, 13, 1, 16, 3...
NumDealsPurchases       [3, 2, 1, 5, 4, 15, 7, 0, 6, 9, 12, 8, 10, 13,...
NumWebPurchases         [8, 1, 2, 5, 6, 7, 4, 3, 11, 0, 27, 10, 9, 23,...
NumCatalogPurchases        [10, 1, 2, 0, 3, 4, 6, 28, 9, 5, 8, 7, 11, 22]
NumStorePurchases          [4, 2, 10, 6, 7, 0, 3, 8, 5, 12, 9, 13, 11, 1]
NumWebVisitsMonth       [7, 5, 4, 6, 8, 9, 20, 2, 3, 1, 10, 0, 14, 19,...
AcceptedCmp3                                                       [0, 1]
AcceptedCmp4                                                       [0, 1]
AcceptedCmp5                                                       [0, 1]
AcceptedCmp1                                                       [0, 1]
AcceptedCmp2                                                       [0, 1]
Complain                                                           [0, 1]
Z_CostContact                                                         [3]
Z_Revenue                                                            [11]
Response                                                           [1, 0]
dtype: object
```

In [11]:
```
dupl_df[dupl_df.apply(pd.Series.nunique, axis=1) == 1]
dupl_df["Complain"].equals(dupl_df["AcceptedCmp3"])
#AcceptedCmp1,2,3,4,5 not related to complain
```

Out[11]:
```
False
```

Checking the conclusion about the cvolumns AcceptedCmp1,2,3,4,5 and response

Assumption- If any of AcceptedCmp1,2,3,4,5 is true then Response is true

In [12]:
```
any_accepted = (dupl_df['AcceptedCmp1'] == 1) | (dupl_df['AcceptedCmp2'] == 1) | (d
response_accp = (dupl_df['Response'] == 1)
result = any_accepted & response_accp
dupl_df[result]
```

Out[12]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cust
15	2114	1946	PhD	Single	82800.0	0	0	2012-1
39	2968	1943	PhD	Divorced	48948.0	0	0	2013-(
53	2225	1977	Graduation	Divorced	82582.0	0	0	2014-(
55	6260	1955	Master	Together	82384.0	0	0	2012-1
60	6853	1982	Master	Single	75777.0	0	0	2013-(
...	...	...	...	...	...	...	...	
2175	1772	1975	PhD	Married	79174.0	0	0	2013-(
2193	8722	1957	2n Cycle	Married	82347.0	0	0	2012-1
2194	7118	1957	Graduation	Married	73803.0	0	1	2012-(
2198	2632	1954	Graduation	Married	50501.0	1	1	2013-(
2221	7366	1982	Master	Single	75777.0	0	0	2013-(

188 rows × 29 columns

In [13]:
```
any_accepted = (dupl_df['AcceptedCmp1'] == 0) | (dupl_df['AcceptedCmp2'] == 0) | (d
response_accp = (dupl_df['Response'] == 1)
result = any_accepted & response_accp
dupl_df[result]
```

Out[13]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
0	5524	1957	Graduation	Single	58138.0	0	0	2012-
8	4855	1974	PhD	Together	30351.0	1	0	2013-
15	2114	1946	PhD	Single	82800.0	0	0	2012-
33	7373	1952	PhD	Divorced	46610.0	0	2	2012-
39	2968	1943	PhD	Divorced	48948.0	0	0	2013-
...	...	...	...	...	...	...	...	
2194	7118	1957	Graduation	Married	73803.0	0	1	2012-
2198	2632	1954	Graduation	Married	50501.0	1	1	2013-
2202	11133	1973	PhD	YOLO	48432.0	0	1	2012-
2221	7366	1982	Master	Single	75777.0	0	0	2013-
2239	9405	1954	PhD	Married	52869.0	1	1	2012-

334 rows × 29 columns

Our Above assumption is wrong as there are 334 rows having AcceptedCmp1,2,3,4,5 all false but Response as True

Now we can assume Response as customer accepted the last offer or not

Data Dictionary

1. ID 2240 - Unique customer ids
2. Year_Birth 59 - Birthdate of customers
3. Education 5 - Education qualification of customers
4. Marital_Status 8 - Married/single/divorced/together
5. Income 1974 - Income of customer
6. Kidhome 3 - no. of Kids in a house(below 13yrs)
7. Teenhome 3 - no. of Teenagers(13 to 19 yrs)
8. Dt_Customer 663 - Date of first purchase/membership enrollment
9. Recency 100 - No. of visits in last 1 year
10. MntWines 776 - Amnt spent on wines
11. MntFruits 158 -Amnt spent on fruits
12. MntMeatProducts 558- Amnt spent on Meat
13. MntFishProducts 182- Amnt spent on fish prod
14. MntSweetProducts 177- Amnt spent on sweet products

15. MntGoldProds 213- Amnt spent on gold products
16. NumDealsPurchases 15 - Number of deals purchased
17. NumWebPurchases 15 - No. of purchases from website of supermarket
18. NumCatalogPurchases 14- No. of purchases from website of catalog
19. NumStorePurchases 14-No. of purchases by offline visit of supermarket
20. NumWebVisitsMonth 16- Avg. No. of visits in a month
21. AcceptedCmp3 2 - Accepted offer in 3rd attempt or not
22. AcceptedCmp4 2 - Accepted offer in 4th attempt or not
23. AcceptedCmp5 2- Accepted offer in 5th attempt or not
24. AcceptedCmp1 2- Accepted offer in single attempt or not
25. AcceptedCmp2 2 - Accepted offer in 2nd attempt or not
26. Complain 2 - 1 for complain raised and for not raised
27. Z_CostContact 1 - Unknown- all duplicate
28. Z_Revenue 1 - Unknown- all duplicate
29. Response 2 - Aceepted the last offer or not

Checking for missing values in dataset

In [14]:
```
dupl_df.isna()
```

Out[14]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custo
0	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	
2235	False	False	False	False	False	False	False	F
2236	False	False	False	False	False	False	False	F
2237	False	False	False	False	False	False	False	F
2238	False	False	False	False	False	False	False	F
2239	False	False	False	False	False	False	False	F

2240 rows × 29 columns

Percentage of missing values in dataset in each column

In [15]:
```
dupl_df.isna().sum()
```

Out[15]:
```
ID                      0
Year_Birth              0
Education               0
Marital_Status          0
Income                 24
Kidhome                 0
Teenhome                0
Dt_Customer             0
Recency                 0
MntWines                0
MntFruits               0
MntMeatProducts         0
MntFishProducts         0
MntSweetProducts        0
MntGoldProds            0
NumDealsPurchases       0
NumWebPurchases         0
NumCatalogPurchases     0
NumStorePurchases       0
NumWebVisitsMonth       0
AcceptedCmp3            0
AcceptedCmp4            0
AcceptedCmp5            0
AcceptedCmp1            0
AcceptedCmp2            0
Complain                0
Z_CostContact           0
Z_Revenue               0
Response                0
dtype: int64
```

In [16]:
```
(dupl_df.isna().sum()/len(dupl_df))*100
```

```
Out[16]: ID                     0.000000
         Year_Birth             0.000000
         Education              0.000000
         Marital_Status         0.000000
         Income                 1.071429
         Kidhome                0.000000
         Teenhome               0.000000
         Dt_Customer            0.000000
         Recency                0.000000
         MntWines               0.000000
         MntFruits              0.000000
         MntMeatProducts        0.000000
         MntFishProducts        0.000000
         MntSweetProducts       0.000000
         MntGoldProds           0.000000
         NumDealsPurchases      0.000000
         NumWebPurchases        0.000000
         NumCatalogPurchases    0.000000
         NumStorePurchases      0.000000
         NumWebVisitsMonth      0.000000
         AcceptedCmp3           0.000000
         AcceptedCmp4           0.000000
         AcceptedCmp5           0.000000
         AcceptedCmp1           0.000000
         AcceptedCmp2           0.000000
         Complain               0.000000
         Z_CostContact          0.000000
         Z_Revenue              0.000000
         Response               0.000000
         dtype: float64
```

Handling missing values and droping duplicate values

```
In [17]: missing_values = dupl_df[eda_df.isnull().any(axis=1)]

         # Display the rows containing missing values
         print(missing_values)
```

```
        ID  Year_Birth   Education Marital_Status  Income  Kidhome  Teenhome  \
10    1994        1983  Graduation        Married     NaN        1         0
27    5255        1986  Graduation         Single     NaN        1         0
43    7281        1959         PhD         Single     NaN        0         0
48    7244        1951  Graduation         Single     NaN        2         1
58    8557        1982  Graduation         Single     NaN        1         0
71   10629        1973    2n Cycle        Married     NaN        1         0
90    8996        1957         PhD        Married     NaN        2         1
91    9235        1957  Graduation         Single     NaN        1         1
92    5798        1973      Master       Together     NaN        0         0
128   8268        1961         PhD        Married     NaN        0         1
133   1295        1963  Graduation        Married     NaN        0         1
312   2437        1989  Graduation        Married     NaN        0         0
319   2863        1970  Graduation         Single     NaN        1         2
1379 10475        1970      Master       Together     NaN        0         1
1382  2902        1958  Graduation       Together     NaN        1         1
1383  4345        1964    2n Cycle         Single     NaN        1         1
1386  3769        1972         PhD       Together     NaN        1         0
2059  7187        1969      Master       Together     NaN        1         1
2061  1612        1981         PhD         Single     NaN        1         0
2078  5079        1971  Graduation        Married     NaN        1         1
2079 10339        1954      Master       Together     NaN        0         1
2081  3117        1955  Graduation         Single     NaN        0         1
2084  5250        1943      Master          Widow     NaN        0         0
2228  8720        1978    2n Cycle       Together     NaN        0         0

     Dt_Customer  Recency  MntWines  ...  NumWebVisitsMonth  AcceptedCmp3  \
10    2013-11-15       11         5  ...                  7             0
27    2013-02-20       19         5  ...                  1             0
43    2013-11-05       80        81  ...                  2             0
48    2014-01-01       96        48  ...                  6             0
58    2013-06-17       57        11  ...                  6             0
71    2012-09-14       25        25  ...                  8             0
90    2012-11-19        4       230  ...                  9             0
91    2014-05-27       45         7  ...                  7             0
92    2013-11-23       87       445  ...                  1             0
128   2013-07-11       23       352  ...                  6             0
133   2013-08-11       96       231  ...                  4             0
312   2013-06-03       69       861  ...                  3             0
319   2013-08-23       67       738  ...                  7             0
1379  2013-04-01       39       187  ...                  5             0
1382  2012-09-03       87        19  ...                  5             0
1383  2014-01-12       49         5  ...                  7             0
1386  2014-03-02       17        25  ...                  7             0
2059  2013-05-18       52       375  ...                  3             0
2061  2013-05-31       82        23  ...                  6             0
2078  2013-03-03       82        71  ...                  8             0
2079  2013-06-23       83       161  ...                  6             0
2081  2013-10-18       95       264  ...                  7             0
2084  2013-10-30       75       532  ...                  1             0
2228  2012-08-12       53        32  ...                  0             0

      AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Complain  \
10               0             0             0             0         0
27               0             0             0             0         0
43               0             0             0             0         0
```

```
48             0          0          0          0       0
58             0          0          0          0       0
71             0          0          0          0       0
90             0          0          0          0       0
91             0          0          0          0       0
92             0          0          0          0       0
128            0          0          0          0       0
133            0          0          0          0       0
312            1          0          1          0       0
319            1          0          1          0       0
1379           0          0          0          0       0
1382           0          0          0          0       0
1383           0          0          0          0       0
1386           0          0          0          0       0
2059           0          0          0          0       0
2061           0          0          0          0       0
2078           0          0          0          0       0
2079           0          0          0          0       0
2081           0          0          0          0       0
2084           0          1          0          0       0
2228           1          0          0          0       0

      Z_CostContact  Z_Revenue  Response
10                3         11         0
27                3         11         0
43                3         11         0
48                3         11         0
58                3         11         0
71                3         11         0
90                3         11         0
91                3         11         0
92                3         11         0
128               3         11         0
133               3         11         0
312               3         11         0
319               3         11         0
1379              3         11         0
1382              3         11         0
1383              3         11         0
1386              3         11         0
2059              3         11         0
2061              3         11         0
2078              3         11         0
2079              3         11         0
2081              3         11         0
2084              3         11         1
2228              3         11         0

[24 rows x 29 columns]
```

Filling NaN values with mean income

In [18]:

```
income_mean = dupl_df['Income'].mean()
dupl_df['Income'].fillna(income_mean, inplace=True)
```

```
dupl_df
```

Out[18]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
0	5524	1957	Graduation	Single	58138.0	0	0	2012-
1	2174	1954	Graduation	Single	46344.0	1	1	2014-
2	4141	1965	Graduation	Together	71613.0	0	0	2013-
3	6182	1984	Graduation	Together	26646.0	1	0	2014-
4	5324	1981	PhD	Married	58293.0	1	0	2014-
...	...	...	...	...	...	...	...	
2235	10870	1967	Graduation	Married	61223.0	0	1	2013-
2236	4001	1946	PhD	Together	64014.0	2	1	2014-
2237	7270	1981	Graduation	Divorced	56981.0	0	0	2014-
2238	8235	1956	Master	Together	69245.0	0	1	2014-
2239	9405	1954	PhD	Married	52869.0	1	1	2012-

2240 rows × 29 columns

In [19]:
```
dupl_df.isna().sum()
```

Out[19]:
```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                 0
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

In [20]:
```
columns_to_drop = ['Z_Revenue', 'Z_CostContact']
dupl_df.drop(columns_to_drop, axis=1, inplace=True)
```

In [21]:
```
dupl_df
```

Out[21]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Cus
0	5524	1957	Graduation	Single	58138.0	0	0	2012
1	2174	1954	Graduation	Single	46344.0	1	1	2014
2	4141	1965	Graduation	Together	71613.0	0	0	2013
3	6182	1984	Graduation	Together	26646.0	1	0	2014
4	5324	1981	PhD	Married	58293.0	1	0	2014
...	...	...	...	...	...	...	...	
2235	10870	1967	Graduation	Married	61223.0	0	1	2013
2236	4001	1946	PhD	Together	64014.0	2	1	2014
2237	7270	1981	Graduation	Divorced	56981.0	0	0	2014
2238	8235	1956	Master	Together	69245.0	0	1	2014
2239	9405	1954	PhD	Married	52869.0	1	1	2012

2240 rows × 27 columns

statistical properties of columns having numerical values as well as at least 5 unique column values.

In [22]:

```
# Duplicating the dataset and then droping the Ids columns as ids has no role in st
dup_sdf=dupl_df.copy()
dup_sdf.drop('ID', axis=1, inplace=True)
```

In [23]:

```
dup_sdf
```

Out[23]:

	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer
0	1957	Graduation	Single	58138.0	0	0	2012-09-04
1	1954	Graduation	Single	46344.0	1	1	2014-03-08
2	1965	Graduation	Together	71613.0	0	0	2013-08-21
3	1984	Graduation	Together	26646.0	1	0	2014-02-10
4	1981	PhD	Married	58293.0	1	0	2014-01-19
...	...	...	...	...	...	...	...
2235	1967	Graduation	Married	61223.0	0	1	2013-06-13
2236	1946	PhD	Together	64014.0	2	1	2014-06-10
2237	1981	Graduation	Divorced	56981.0	0	0	2014-01-25
2238	1956	Master	Together	69245.0	0	1	2014-01-24
2239	1954	PhD	Married	52869.0	1	1	2012-10-15

2240 rows × 26 columns

In [24]:

```
filtered_columns = dup_sdf.columns[dup_sdf.nunique() >= 5]

# Calculate statistical properties for the filtered columns
statistics = dup_sdf[filtered_columns].describe()
statistics
```

Out[24]:

	Year_Birth	Income	Dt_Customer	Recency	MntWines	MntFru
count	2240.000000	2240.000000	2240	2240.000000	2240.000000	2240.0000
mean	1968.805804	52247.251354	2013-07-10 10:01:42.857142784	49.109375	303.935714	26.3022
min	1893.000000	1730.000000	2012-07-30 00:00:00	0.000000	0.000000	0.0000
25%	1959.000000	35538.750000	2013-01-16 00:00:00	24.000000	23.750000	1.0000
50%	1970.000000	51741.500000	2013-07-08 12:00:00	49.000000	173.500000	8.0000
75%	1977.000000	68289.750000	2013-12-30 06:00:00	74.000000	504.250000	33.0000
max	1996.000000	666666.000000	2014-06-29 00:00:00	99.000000	1493.000000	199.0000
std	11.984069	25037.797168	NaN	28.962453	336.597393	39.7734

Correlation

In [25]:
```
##Slicing Id as it doesnt effect correlation and columns having strings/object
dupl_df.iloc[:, ~dupl_df.columns.isin(dupl_df.columns[[0, 2, 3]])]
```

Out[25]:

	Year_Birth	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	MntFr
0	1957	58138.0	0	0	2012-09-04	58	635	
1	1954	46344.0	1	1	2014-03-08	38	11	
2	1965	71613.0	0	0	2013-08-21	26	426	
3	1984	26646.0	1	0	2014-02-10	26	11	
4	1981	58293.0	1	0	2014-01-19	94	173	
...	...	...	...	...	...	...	...	
2235	1967	61223.0	0	1	2013-06-13	46	709	
2236	1946	64014.0	2	1	2014-06-10	56	406	
2237	1981	56981.0	0	0	2014-01-25	91	908	
2238	1956	69245.0	0	1	2014-01-24	8	428	
2239	1954	52869.0	1	1	2012-10-15	40	84	

2240 rows × 24 columns

In [26]:
```
dupl_df.iloc[:, ~dupl_df.columns.isin(dupl_df.columns[[0, 2, 3]])].corr()
```

Out[26]:

	Year_Birth	Income	Kidhome	Teenhome	Dt_Customer	Recency
Year_Birth	1.000000	-0.160942	0.230176	-0.352111	-0.022431	-0.019871
Income	-0.160942	1.000000	-0.425176	0.019018	0.018460	-0.003946
Kidhome	0.230176	-0.425176	1.000000	-0.036133	0.053343	0.008827
Teenhome	-0.352111	0.019018	-0.036133	1.000000	-0.017465	0.016198
Dt_Customer	-0.022431	0.018460	0.053343	-0.017465	1.000000	-0.024522
Recency	-0.019871	-0.003946	0.008827	0.016198	-0.024522	1.000000
MntWines	-0.157773	0.576789	-0.496297	0.004846	-0.166264	0.016064
MntFruits	-0.017917	0.428747	-0.372581	-0.176764	-0.066928	-0.004306
MntMeatProducts	-0.030872	0.577802	-0.437129	-0.261160	-0.092713	0.023056
MntFishProducts	-0.041625	0.437497	-0.387644	-0.204187	-0.080769	0.001079
MntSweetProducts	-0.018133	0.436162	-0.370673	-0.162475	-0.081268	0.022670
MntGoldProds	-0.061818	0.321978	-0.349595	-0.021725	-0.159596	0.016693
NumDealsPurchases	-0.060846	-0.082290	0.221798	0.387741	-0.218552	-0.001098
NumWebPurchases	-0.145040	0.380550	-0.361647	0.155500	-0.191876	-0.010726
NumCatalogPurchases	-0.121275	0.586725	-0.502237	-0.110769	-0.096198	0.025110
NumStorePurchases	-0.128272	0.526489	-0.499683	0.050695	-0.110592	0.000799
NumWebVisitsMonth	0.121139	-0.549824	0.447846	0.134884	-0.272449	-0.021445
AcceptedCmp3	0.061774	-0.016168	0.014674	-0.042677	0.007713	-0.032991
AcceptedCmp4	-0.060510	0.182791	-0.161600	0.038886	-0.018426	0.018826
AcceptedCmp5	0.007123	0.334850	-0.205634	-0.191050	0.005918	0.000129
AcceptedCmp1	-0.005930	0.274921	-0.172339	-0.140090	0.039569	-0.019283
AcceptedCmp2	-0.006539	0.087538	-0.081716	-0.015605	-0.006064	-0.001781
Complain	-0.030128	-0.027223	0.040207	0.003138	-0.033120	0.013231
Response	0.021325	0.132756	-0.080008	-0.154446	-0.194481	-0.198437

24 rows × 24 columns

By analysing the above correlations we can conclude various things, Insights:

For eg: Amount invested on wines is positively correlated to income- Those having more income are used to invest more on wines.

And is negatively correlated to kidhome means those having kids of upto 12yrs are less prone to wines.

Complain is positively correlated with kidhome and teenhome, hence we can conclude that families having children are having more complains than the others, need to improve the products related to kids and teenagers

Visualizations

In [27]:
```
dupl_df['Marital_Status'].value_counts()
```

Out[27]:
```
Marital_Status
Married     864
Together    580
Single      480
Divorced    232
Widow        77
Alone         3
Absurd        2
YOLO          2
Name: count, dtype: int64
```

In [28]:
```
#Combining maritial status
dupl_df['Marital_Status'] = dupl_df['Marital_Status'].replace(['Married', 'Together
dupl_df['Marital_Status'] = dupl_df['Marital_Status'].replace(['Divorced', 'Widow',
```

In [29]:
```
dupl_df['Marital_Status'].value_counts()
```

Out[29]:
```
Marital_Status
relationship    1444
Single           796
Name: count, dtype: int64
```

In [30]:
```
# Combining columns
dupl_df['Kids'] = dupl_df['Kidhome'] + dupl_df['Teenhome']
dupl_df['Expenses'] = dupl_df['MntWines'] + dupl_df['MntFruits'] + dupl_df['MntMeat
dupl_df['TotalAcceptedCmp'] = dupl_df['AcceptedCmp1'] + dupl_df['AcceptedCmp2'] + d
dupl_df['NumTotalPurchases'] = dupl_df['NumWebPurchases'] + dupl_df['NumCatalogPurc
```

In [31]:
```
#Dropping remaining columns
dupl_df=dupl_df.drop(columns=["AcceptedCmp1" , "AcceptedCmp2", "AcceptedCmp3" , "Ac
```

In [32]:
```
dupl_df
```

Out[32]:

	ID	Year_Birth	Education	Marital_Status	Income	Dt_Customer	Recency	Com
0	5524	1957	Graduation	Single	58138.0	2012-09-04	58	
1	2174	1954	Graduation	Single	46344.0	2014-03-08	38	
2	4141	1965	Graduation	relationship	71613.0	2013-08-21	26	
3	6182	1984	Graduation	relationship	26646.0	2014-02-10	26	
4	5324	1981	PhD	relationship	58293.0	2014-01-19	94	
...	...	...	...	...	...	...	...	
2235	10870	1967	Graduation	relationship	61223.0	2013-06-13	46	
2236	4001	1946	PhD	relationship	64014.0	2014-06-10	56	
2237	7270	1981	Graduation	Single	56981.0	2014-01-25	91	
2238	8235	1956	Master	relationship	69245.0	2014-01-24	8	
2239	9405	1954	PhD	relationship	52869.0	2012-10-15	40	

2240 rows × 12 columns

In [33]:
```
#AGE
dupl_df["Age"]=2021 - dupl_df["Year_Birth"]
```

In [34]:
```
dupl_df['Education'].value_counts()
dupl_df['Education'] = dupl_df['Education'].replace(['PhD','2n Cycle','Graduation',
dupl_df['Education'] = dupl_df['Education'].replace(['Basic'], 'UG')
```

In [35]:
```
dupl_df['Years_Completed'] = 2021- dupl_df['Dt_Customer'].dt.year
```

In [36]:
```
dupl_df=dupl_df.drop(columns=["ID","Year_Birth","Dt_Customer","Complain"],axis=1)
```

In [37]:
```
dupl_df
```

Out[37]:

	Education	Marital_Status	Income	Recency	Kids	Expenses	TotalAcceptedCmp	Nu
0	PG	Single	58138.0	58	0	1617	1	
1	PG	Single	46344.0	38	2	27	0	
2	PG	relationship	71613.0	26	0	776	0	
3	PG	relationship	26646.0	26	1	53	0	
4	PG	relationship	58293.0	94	1	422	0	
...	...	...	...	...	...	...	...	
2235	PG	relationship	61223.0	46	1	1341	0	
2236	PG	relationship	64014.0	56	3	444	1	
2237	PG	Single	56981.0	91	0	1241	1	
2238	PG	relationship	69245.0	8	1	843	0	
2239	PG	relationship	52869.0	40	2	172	1	

2240 rows × 10 columns

In [38]:
```
dupl_df.iloc[: , 2 :].describe()
```

Out[38]:

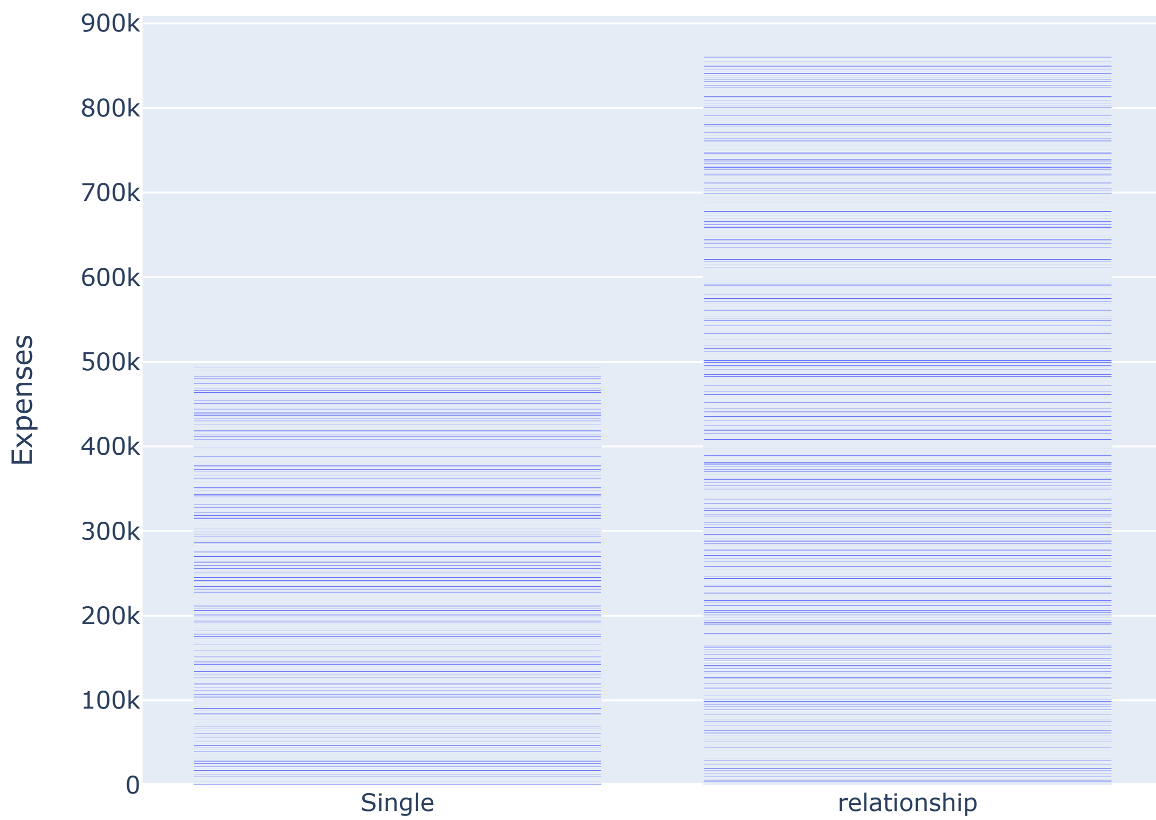
	Income	Recency	Kids	Expenses	TotalAcceptedCmp	NumTota
count	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2
mean	52247.251354	49.109375	0.950446	605.798214	0.446875	
std	25037.797168	28.962453	0.751803	602.249288	0.890543	
min	1730.000000	0.000000	0.000000	5.000000	0.000000	
25%	35538.750000	24.000000	0.000000	68.750000	0.000000	
50%	51741.500000	49.000000	1.000000	396.000000	0.000000	
75%	68289.750000	74.000000	1.000000	1045.500000	1.000000	
max	666666.000000	99.000000	3.000000	2525.000000	5.000000	

In [39]:
```
dupl_df.iloc[: , 2 :].corr()
```

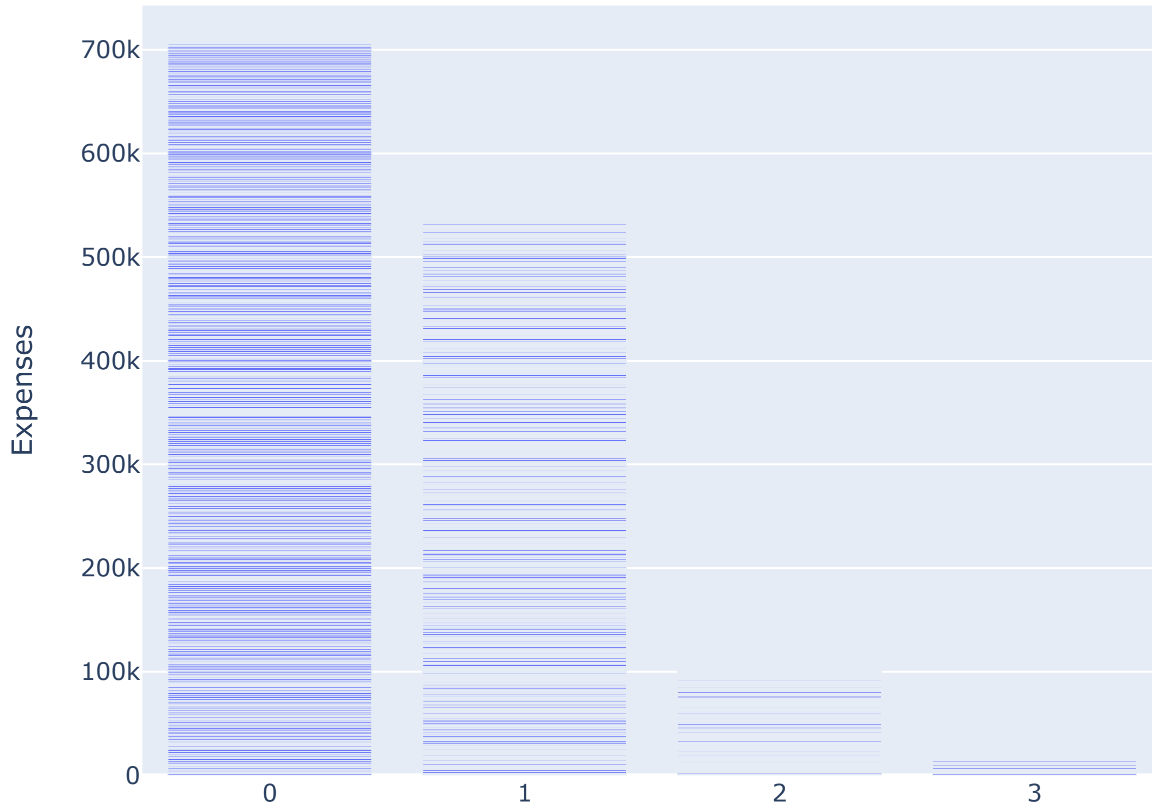
Out[39]:

	Income	Recency	Kids	Expenses	TotalAcceptedCmp	NumTo
Income	1.000000	-0.003946	-0.290712	0.664706	0.287046	
Recency	-0.003946	1.000000	0.018053	0.020433	-0.088962	
Kids	-0.290712	0.018053	1.000000	-0.498888	-0.253760	
Expenses	0.664706	0.020433	-0.498888	1.000000	0.456206	
TotalAcceptedCmp	0.287046	-0.088962	-0.253760	0.456206	1.000000	
NumTotalPurchases	0.563370	0.005740	-0.245790	0.753903	0.258045	
Age	0.160942	0.019871	0.090199	0.111306	-0.008302	
Years_Completed	-0.022366	0.026084	-0.032215	0.144235	0.052129	

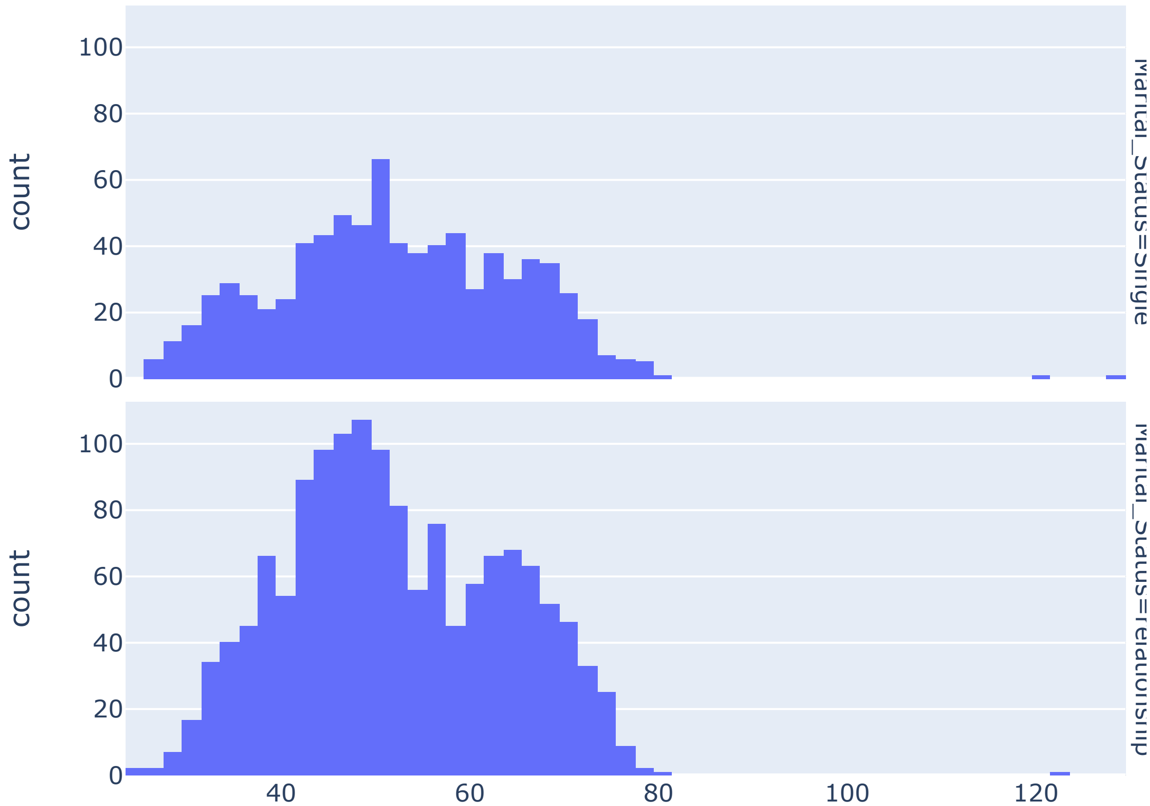
In [40]:
```
import plotly.express as px
px.bar(dupl_df, x='Marital_Status', y='Expenses')
```



In [41]:
```
px.bar(dupl_df, x='Kids', y='Expenses')
```



In [42]:
```
px.histogram(dupl_df,x="Age", facet_row = "Marital_Status")
```

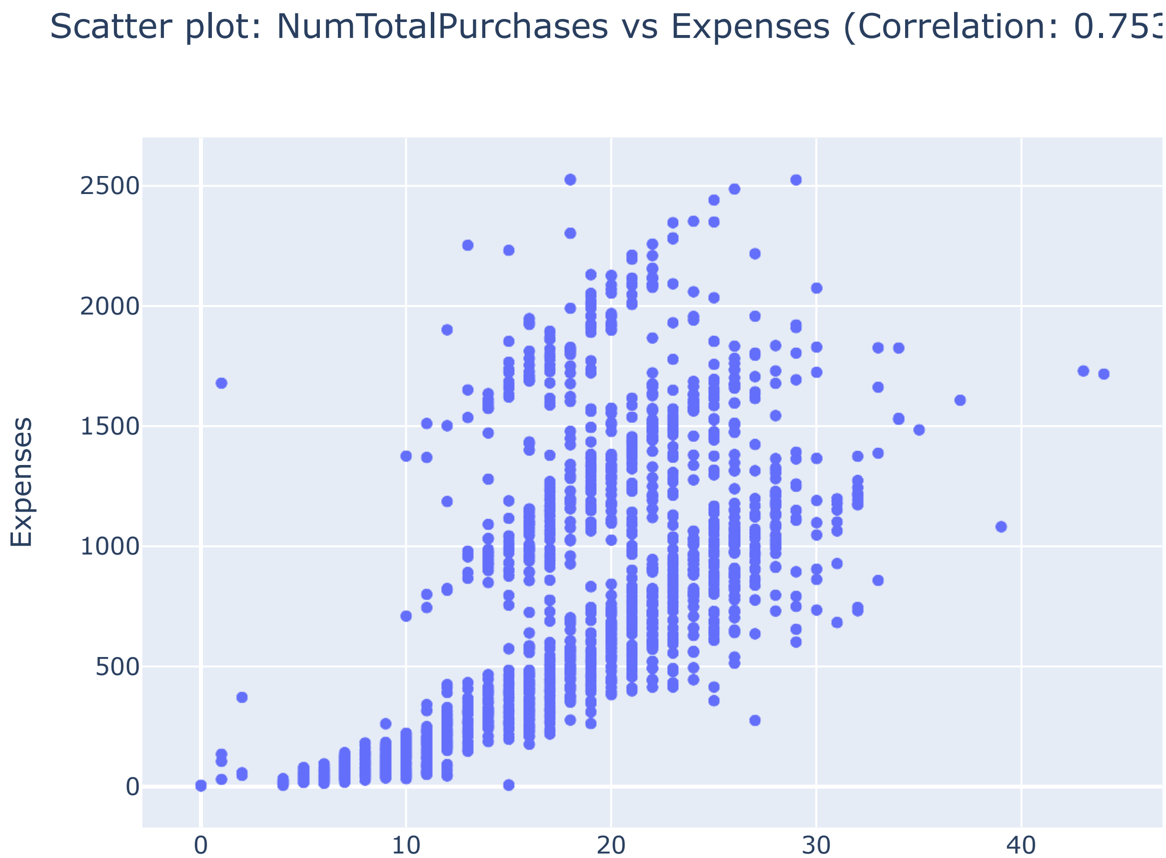


Pair plot for numerical columns only

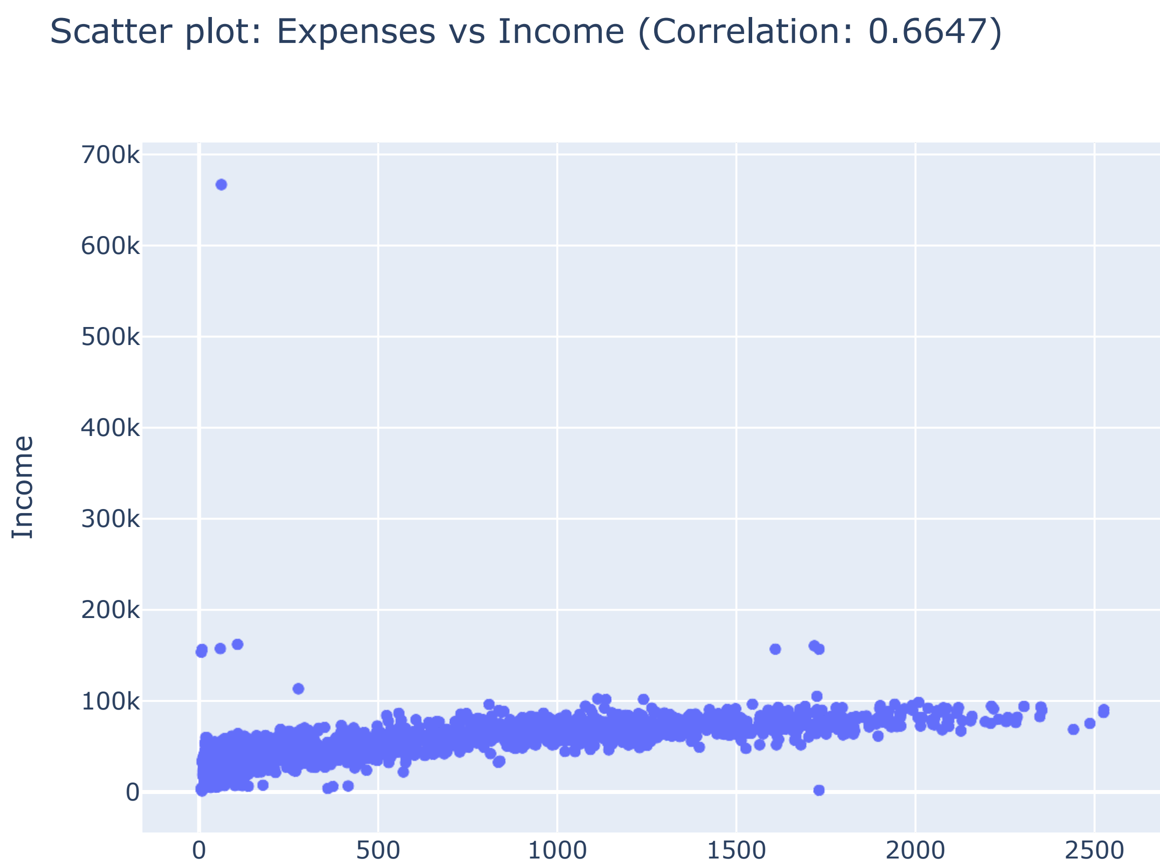
In [43]:
```
numerical_cols = dupl_df.select_dtypes(include='number').columns
px.scatter_matrix(dupl_df[numerical_cols])
```

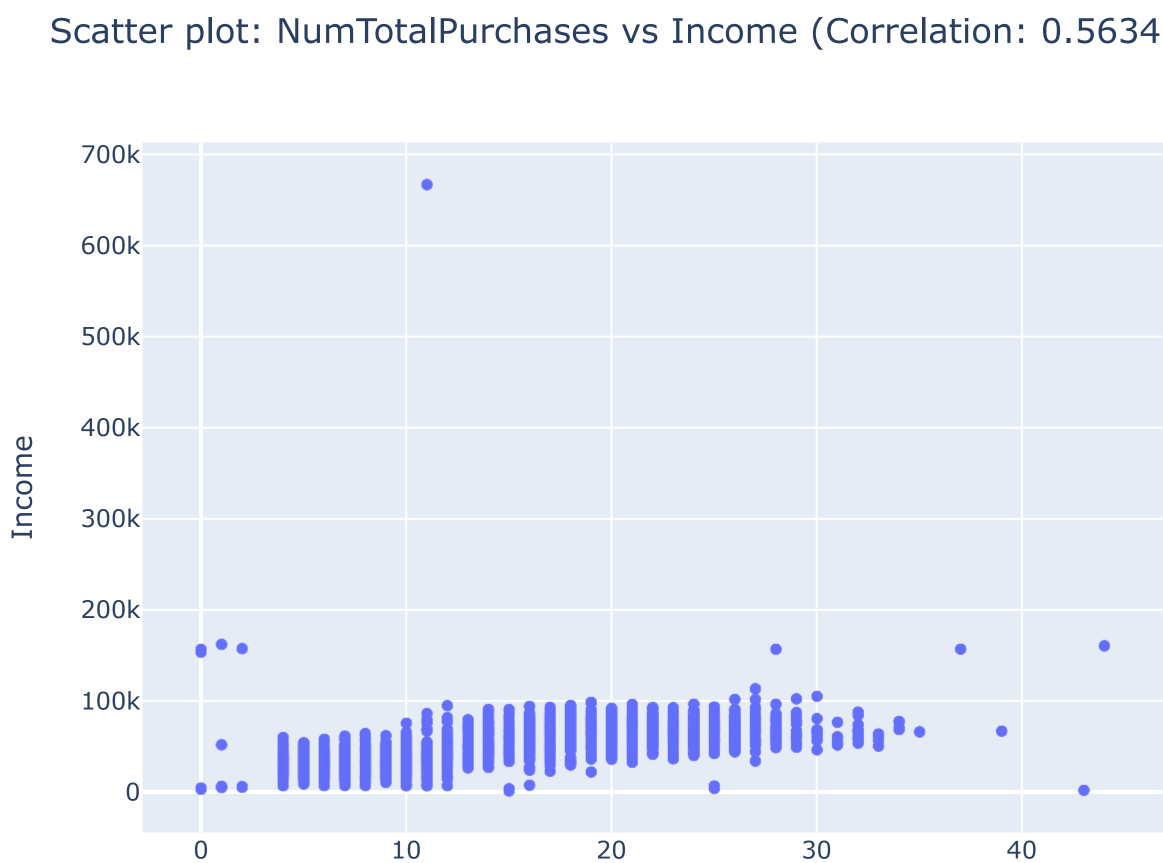
8.

In [44]:
```
correlation_matrix = dupl_df[numerical_cols].corr()

# Find top 3 positive correlation pairs
top_3_positive_corr = correlation_matrix.unstack().sort_values(ascending=False)
top_3_positive_corr = top_3_positive_corr[top_3_positive_corr < 1].drop_duplicates(

# Create scatter plots for the top 3 positive correlation pairs using Plotly
for i in range(0, len(top_3_positive_corr)):
    col1, col2 = top_3_positive_corr.index[i]
    corr_value = top_3_positive_corr[col1, col2]
    fig = px.scatter(dupl_df, x=col1, y=col2, title=f"Scatter plot: {col1} vs {col2
    fig.show()
```

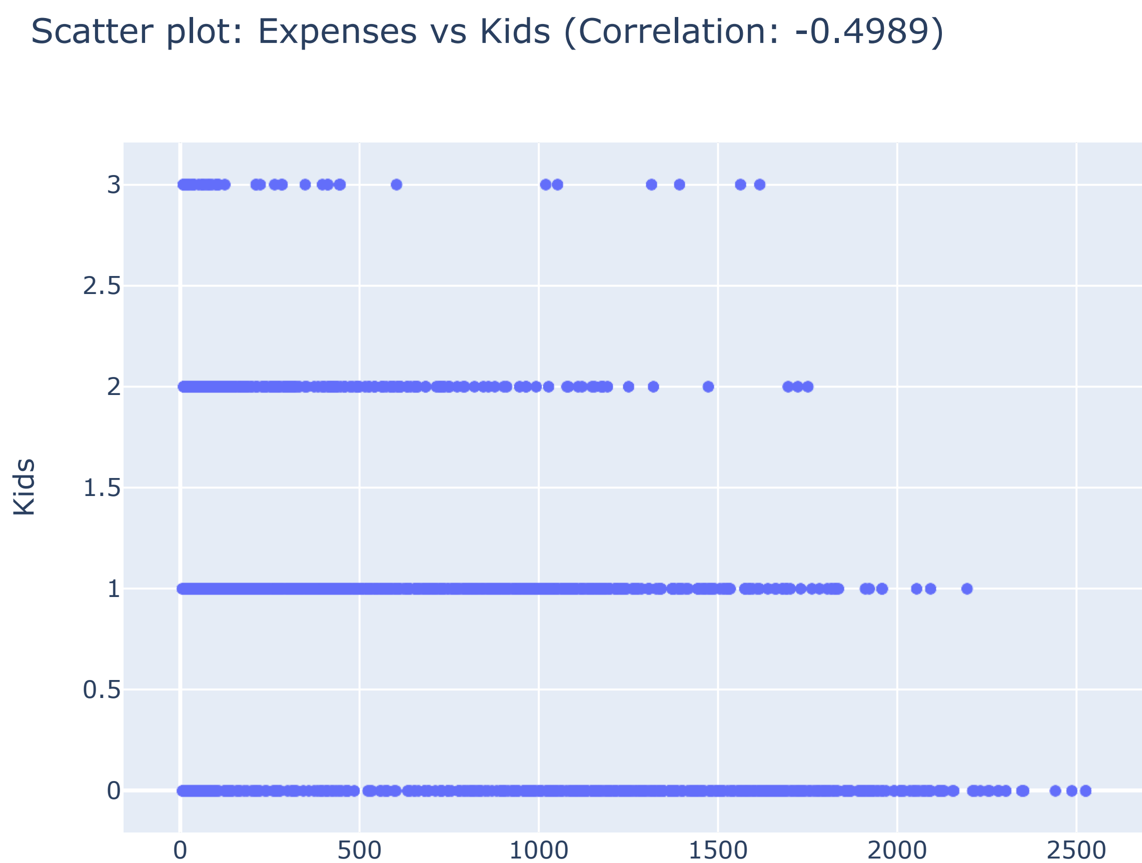


Scatter plot: NumTotalPurchases vs Expenses (Correlation: 0.753
Expenses
2500
2000
1500
1000
500
0
0
10
20
30
40


Scatter plot: Expenses vs Income (Correlation: 0.6647)
Income
700k
600k
500k
400k
300k
200k
100k
0
0
500
1000
1500
2000
2500

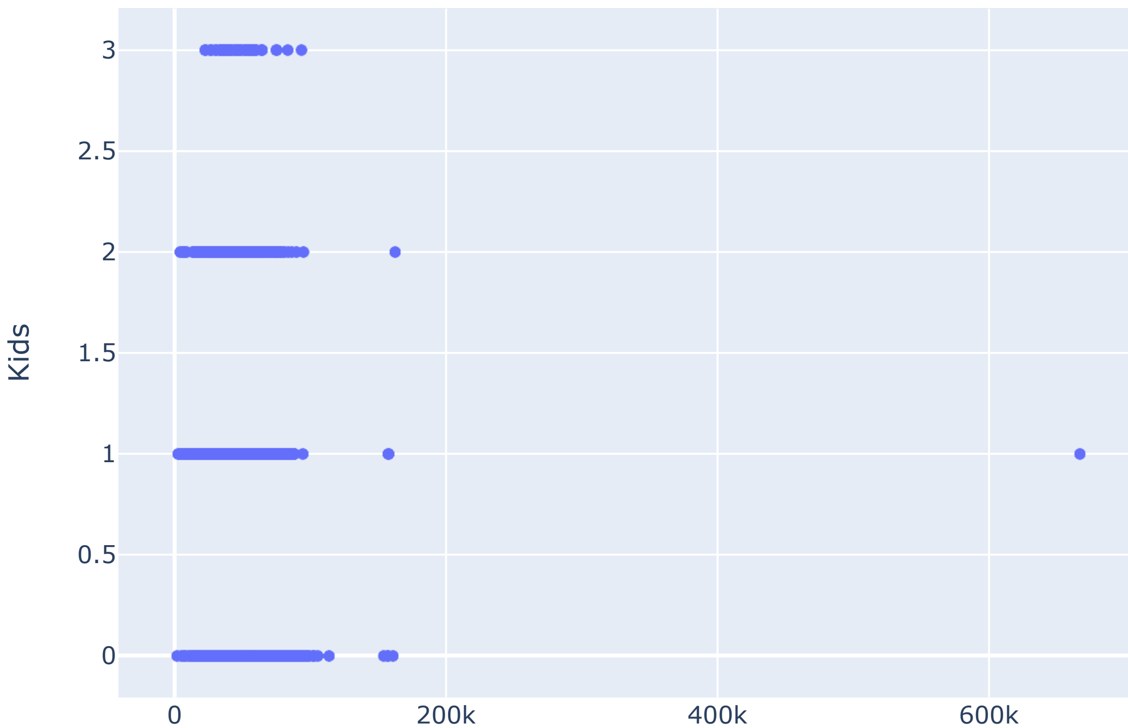


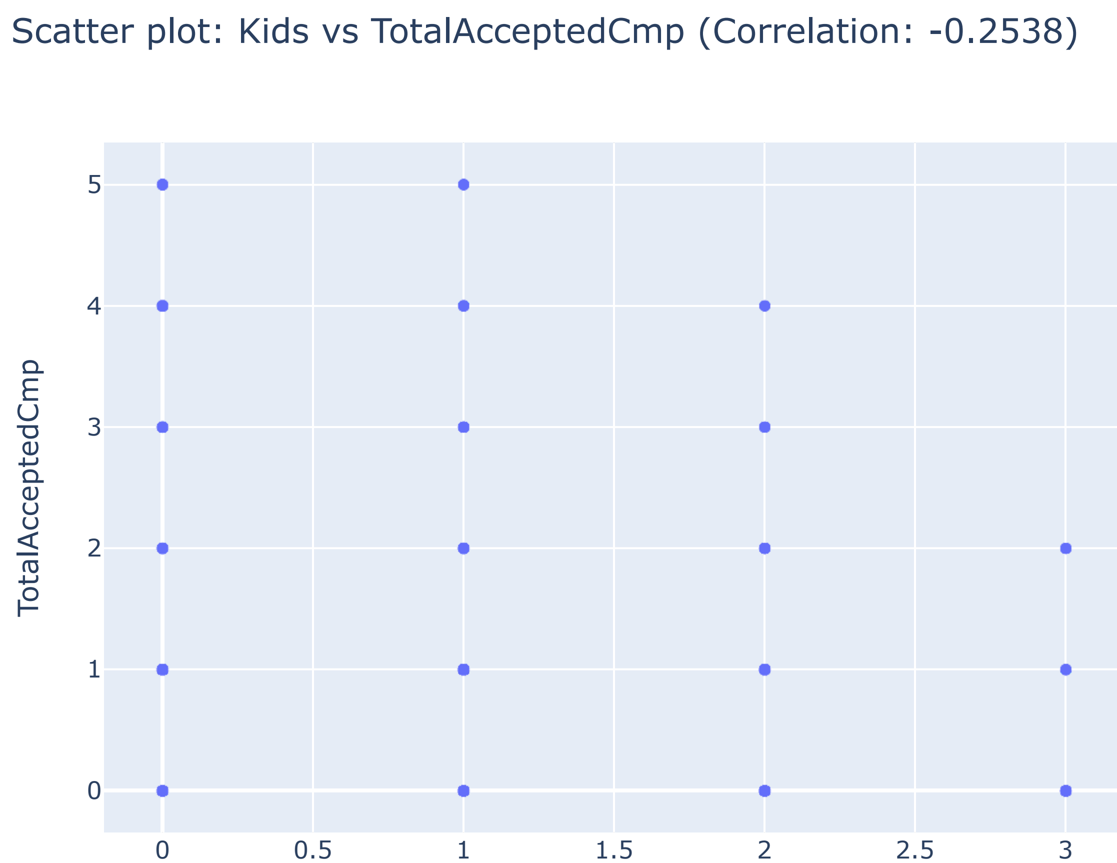
In [45]:
```
correlation_matrix = dupl_df[numerical_cols].corr()

# Find top 3 negative correlation pairs
top_3_positive_corr = correlation_matrix.unstack().sort_values(ascending=True)
top_3_positive_corr = top_3_positive_corr[top_3_positive_corr < 0].drop_duplicates(

# Create scatter plots for the top 3 negative correlation pairs using Plotly
for i in range(0, len(top_3_positive_corr)):
    col1, col2 = top_3_positive_corr.index[i]
    corr_value = top_3_positive_corr[col1, col2]
    fig = px.scatter(dupl_df, x=col1, y=col2, title=f"Scatter plot: {col1} vs {col2
    fig.show()
```



Scatter plot: Expenses vs Kids (Correlation: -0.4989)
Kids
3
2.5
2
1.5
1
0.5
0
0
500
1000
1500
2000
2500

Scatter plot: Income vs Kids (Correlation: -0.2907)




Scatter plot: Kids vs TotalAcceptedCmp (Correlation: -0.2538)
TotalAcceptedCmp
0
1
2
3
4
5
0
0.5
1
1.5
2
2.5
3