Understand Data

Mach dich zu Beginn mit den Daten vertraut, damit du später bei der Datenbereinigung und -aufbereitung weißt, worauf du achten solltest.

Tipp: Die Spalte `'PurchDate'` ist als Unix-Timestamp in ganzen Sekunden angegeben. Um diese Spalte mit `pandas` in ein Datumsformat umzuwandeln, kannst du folgenden Code nutzen: `my_df.loc[:, 'PurchDate'] = pd.to_datetime(my_df.loc[:, 'PurchDate'], unit='s')`

In [3]:
```
#check size
df.shape
```

Out[3]:
```
(65620, 33)
```

In [4]:
```
#check for datatypes
df.dtypes
```

Out[4]:
```
IsBadBuy                            int64
PurchDate                           int64
Auction                            object
VehYear                             int64
VehicleAge                          int64
Make                               object
Model                              object
Trim                               object
SubModel                           object
Color                              object
Transmission                       object
WheelTypeID                       float64
WheelType                          object
VehOdo                              int64
Nationality                        object
Size                               object
TopThreeAmericanName               object
MMRAcquisitionAuctionAveragePrice  float64
MMRAcquisitionAuctionCleanPrice    float64
MMRAcquisitionRetailAveragePrice   float64
MMRAcquisitonRetailCleanPrice      float64
MMRCurrentAuctionAveragePrice      float64
MMRCurrentAuctionCleanPrice        float64
MMRCurrentRetailAveragePrice       float64
MMRCurrentRetailCleanPrice         float64
PRIMEUNIT                          object
AUCGUART                           object
BYRNO                               int64
VNZIP1                              int64
VNST                               object
VehBCost                          float64
IsOnlineSale                        int64
WarrantyCost                        int64
dtype: object
```

In [5]:
```
#check missing values

nan_sum= [df[col].isna().sum() for col in df.columns]
pd.DataFrame(nan_sum, index=df.columns, columns=['Count of NaN-Werte'])
```

Out[5]:

	Count of NaN-Werte
IsBadBuy	0
PurchDate	0
Auction	0
VehYear	0
VehicleAge	0
Make	0
Model	0
Trim	2098
SubModel	7
Color	7
Transmission	8
WheelTypeID	2873
WheelType	2877
VehOdo	0
Nationality	4
Size	4
TopThreeAmericanName	4
MMRAcquisitionAuctionAveragePrice	18
MMRAcquisitionAuctionCleanPrice	18
MMRAcquisitionRetailAveragePrice	18
MMRAcquisitonRetailCleanPrice	18
MMRCurrentAuctionAveragePrice	290
MMRCurrentAuctionCleanPrice	290
MMRCurrentRetailAveragePrice	290
MMRCurrentRetailCleanPrice	290
PRIMEUNIT	62534
AUCGUART	62534
BYRNO	0
VNZIP1	0
VNST	0
VehBCost	56
IsOnlineSale	0
WarrantyCost	0

In [6]:
```
# reminder for later
#PRIMEUNIT and AUCGUART sollten aufgrund der Anzahl der Nan-Werte gelöscht werden.
```

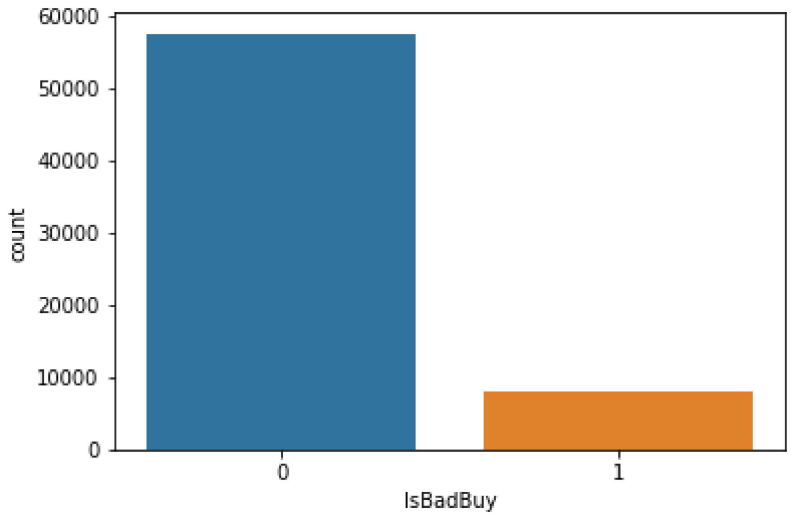
In [7]:
```
#Doppelte Einträge finden

duplicates = df[df.duplicated()]
print(duplicates)

#df = df.drop_duplicates()
#keine doppelten Werte
```

```
Empty DataFrame
Columns: [IsBadBuy, PurchDate, Auction, VehYear, VehicleAge, Make, Model, Trim, Su
bModel, Color, Transmission, WheelTypeID, WheelType, VehOdo, Nationality, Size, To
pThreeAmericanName, MMRAcquisitionAuctionAveragePrice, MMRAcquisitionAuctionCleanP
rice, MMRAcquisitionRetailAveragePrice, MMRAcquisitonRetailCleanPrice, MMRCurrentA
uctionAveragePrice, MMRCurrentAuctionCleanPrice, MMRCurrentRetailAveragePrice, MMR
CurrentRetailCleanPrice, PRIMEUNIT, AUCGUART, BYRNO, VNZIP1, VNST, VehBCost, IsOnl
ineSale, WarrantyCost]
Index: []
```

In [8]:
```
#target variable

df['IsBadBuy'].value_counts() #unausgeglichene Zieldaten
```

Out[8]:
```
0    57516
1     8104
Name: IsBadBuy, dtype: int64
```

In [9]:
```
#Visualization of the target variable

sns.countplot(x="IsBadBuy", data=df)
plt.show()
```



In [10]:
```
#reminder for later
#Die 'PurchDate'-Spalte soll in ein Datumsformat umgewandelt werden.
#df.loc[:, 'PurchDate'] = pd.to_datetime(df.loc[:, 'PurchDate'], unit='s')
```

In [11]:
```
#check unique values
df.nunique()
```

Out[11]:

```
IsBadBuy                              2
PurchDate                           517
Auction                               3
VehYear                              10
VehicleAge                           10
Make                                 29
Model                              1029
Trim                                134
SubModel                            840
Color                                16
Transmission                          3
WheelTypeID                           4
WheelType                             3
VehOdo                            37791
Nationality                           4
Size                                 12
TopThreeAmericanName                  4
MMRAcquisitionAuctionAveragePrice    10147
MMRAcquisitionAuctionCleanPrice      11112
MMRAcquisitionRetailAveragePrice     12445
MMRAcquisitonRetailCleanPrice        13148
MMRCurrentAuctionAveragePrice        10097
MMRCurrentAuctionCleanPrice          11014
MMRCurrentRetailAveragePrice         12233
MMRCurrentRetailCleanPrice           12877
PRIMEUNIT                             2
AUCGUART                              2
BYRNO                                71
VNZIP1                              148
VNST                                 34
VehBCost                           1989
IsOnlineSale                          2
WarrantyCost                        278
dtype: int64
```

In [12]:

```
df[['Auction','VehYear','VehicleAge','Make',
      'Color','Transmission','WheelTypeID','WheelType',
     'Nationality','Size','TopThreeAmericanName','BYRNO','VNST','IsOnlineSale']].ag
```

Out[12]:

	unique
Auction	[OTHER, ADESA, MANHEIM]
VehYear	[2007, 2005, 2006, 2004, 2002, 2003, 2008, 200...
VehicleAge	[2, 4, 5, 3, 6, 7, 8, 1, 9, 0]
Make	[KIA, SUZUKI, CHEVROLET, CHRYSLER, NISSAN, MIT...
Color	[BLACK, SILVER, RED, WHITE, GOLD, GREY, BLUE, ...
Transmission	[MANUAL, AUTO, nan, Manual]
WheelTypeID	[1.0, 2.0, nan, 3.0, 0.0]
WheelType	[Alloy, Covers, nan, Special]
Nationality	[OTHER ASIAN, AMERICAN, TOP LINE ASIAN, OTHER,...
Size	[MEDIUM, COMPACT, VAN, MEDIUM SUV, SPECIALTY, ...
TopThreeAmericanName	[OTHER, GM, CHRYSLER, FORD, nan]
BYRNO	[5546, 20207, 1235, 20928, 835, 21053, 17675, ...
VNST	[AL, TX, UT, FL, AZ, CA, NC, SC, PA, OK, MD, T...
IsOnlineSale	[0, 1]

In [13]:
```
#NOTIZEN:
#IsBadBuy--< 0-1
#PurchDate--> 517 unique werte
#Auction--> Other,Adesa,Manheim--> 3 unique--> Label Encoding
#Vehyear/VahicleAge--> 10 unique Werte
#Make--> 29 unique Werte
#Model-Trim-Submodel-->(1029-134-840 unique) lange kategorische Werte
#Color-->16 unique
#Transmission-->MANUAL,AUTO,Manual--> Textformatierung standardisieren (Manual-MANU
#WheelTypeID-->0-1-2-3
#WheelType--> Alloy,Covers,Special-->3 unique Werte-->Label Encoding
#VehOdo-->normale Verteilung-37791 unique
#Nationality-->OTHER ASIAN; AMERICAN; TOP LINE ASIAN; OTHER-->One Hot Encoding
#Size--> 12 unique Werte-->One Hot Encoding
#TopThreeAmericanName--> OTHER; GM; CHRYSLER;FORD--> One Hot Encoding
#BYRNO--> 71 unique Werte, ID-No
#VNZIP1--> Posleitzahl-->148 unique
#VNST-->Staat--> 34 unique
#VehBCost--> 1989 unique
#IsOnlineSale-->0-1
#WarrantyCost--> 278 unique
```

In [14]:
```
#check categorical and numeric columns
categorical_columns = list(df.select_dtypes(include='object').columns)
numeric_columns = list(df.select_dtypes(include='number').columns)

len(categorical_columns)+ len(numeric_columns) ==len(df.columns)
```

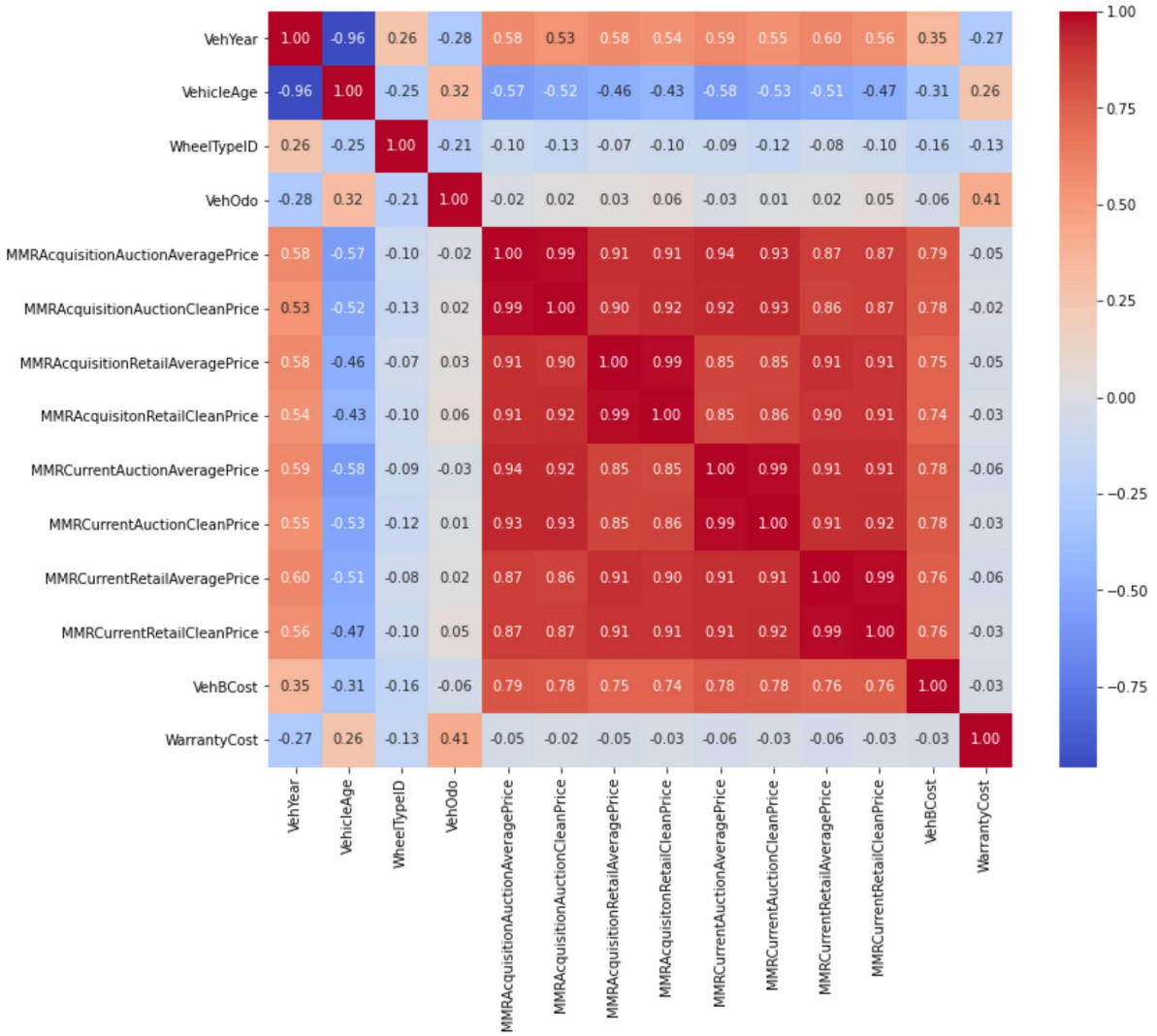
Out[14]:
```
True
```

In [15]:
```
#check describe for numeric columns
# Verteilung der Werte in numerischen Spalten
cols= ['VehYear',
 'VehicleAge',
 'WheelTypeID',
 'VehOdo',
```

```
 'MMRAcquisitionAuctionAveragePrice',
 'MMRAcquisitionAuctionCleanPrice',
 'MMRAcquisitionRetailAveragePrice',
 'MMRAcquisitonRetailCleanPrice',
 'MMRCurrentAuctionAveragePrice',
 'MMRCurrentAuctionCleanPrice',
 'MMRCurrentRetailAveragePrice',
 'MMRCurrentRetailCleanPrice',
 'VehBCost',
 'WarrantyCost']

display(df.loc[:,cols].describe().T)
#display(sns.pairplot(numeric_columns)) -->nicht verständlich
```

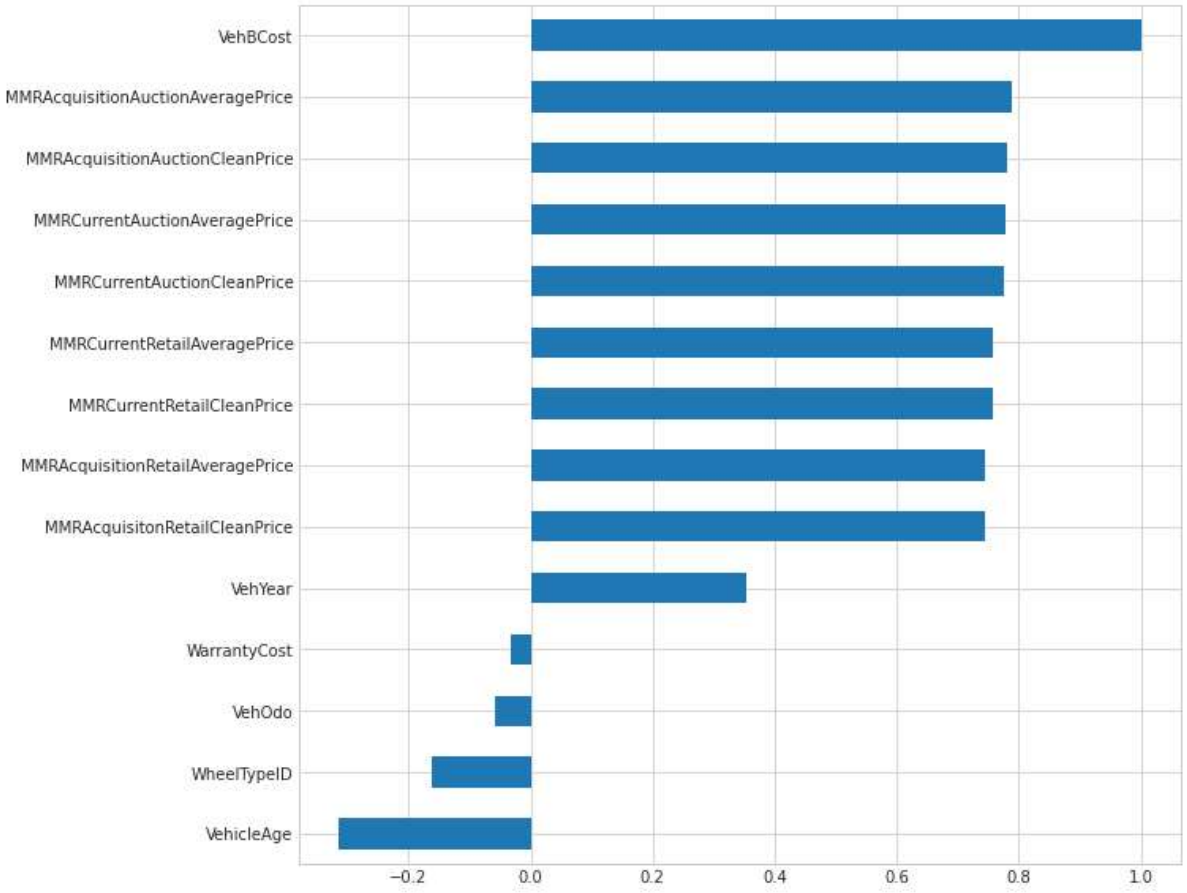
	count	mean	std	min	25%	50
VehYear	65620.0	2005.345215	1.730096	2001.0	2004.00	2005.
VehicleAge	65620.0	4.175130	1.709897	0.0	3.00	4.
WheelTypeID	62747.0	1.494589	0.520986	0.0	1.00	1.
VehOdo	65620.0	71505.930280	14582.977355	5368.0	61844.75	73378.
MMRAcquisitionAuctionAveragePrice	65602.0	6127.143807	2456.965693	0.0	4273.00	6094.
MMRAcquisitionAuctionCleanPrice	65602.0	7371.157236	2716.442814	0.0	5405.00	7303.
MMRAcquisitionRetailAveragePrice	65602.0	8498.240770	3153.983067	0.0	6279.00	8447.
MMRAcquisitonRetailCleanPrice	65602.0	9851.611003	3382.364048	0.0	7494.00	9792.
MMRCurrentAuctionAveragePrice	65330.0	6131.694092	2430.506847	0.0	4275.00	6062.
MMRCurrentAuctionCleanPrice	65330.0	7389.951768	2681.441548	0.0	5412.00	7314.
MMRCurrentRetailAveragePrice	65330.0	8776.866019	3087.008256	0.0	6540.00	8736.
MMRCurrentRetailCleanPrice	65330.0	10145.912582	3305.658577	0.0	7790.25	10103.
VehBCost	65564.0	6727.464264	1759.011960	1.0	5435.00	6700.
WarrantyCost	65620.0	1277.375815	599.971059	462.0	837.00	1169.

In [16]:
```
#heatmap

correlation_matrix = df[cols].corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.show()
```



In [17]:
```
plt.style.use("seaborn-whitegrid")
plt.figure(figsize=(10,10))
df[cols].corr()["VehBCost"].sort_values().plot(kind="barh")
```

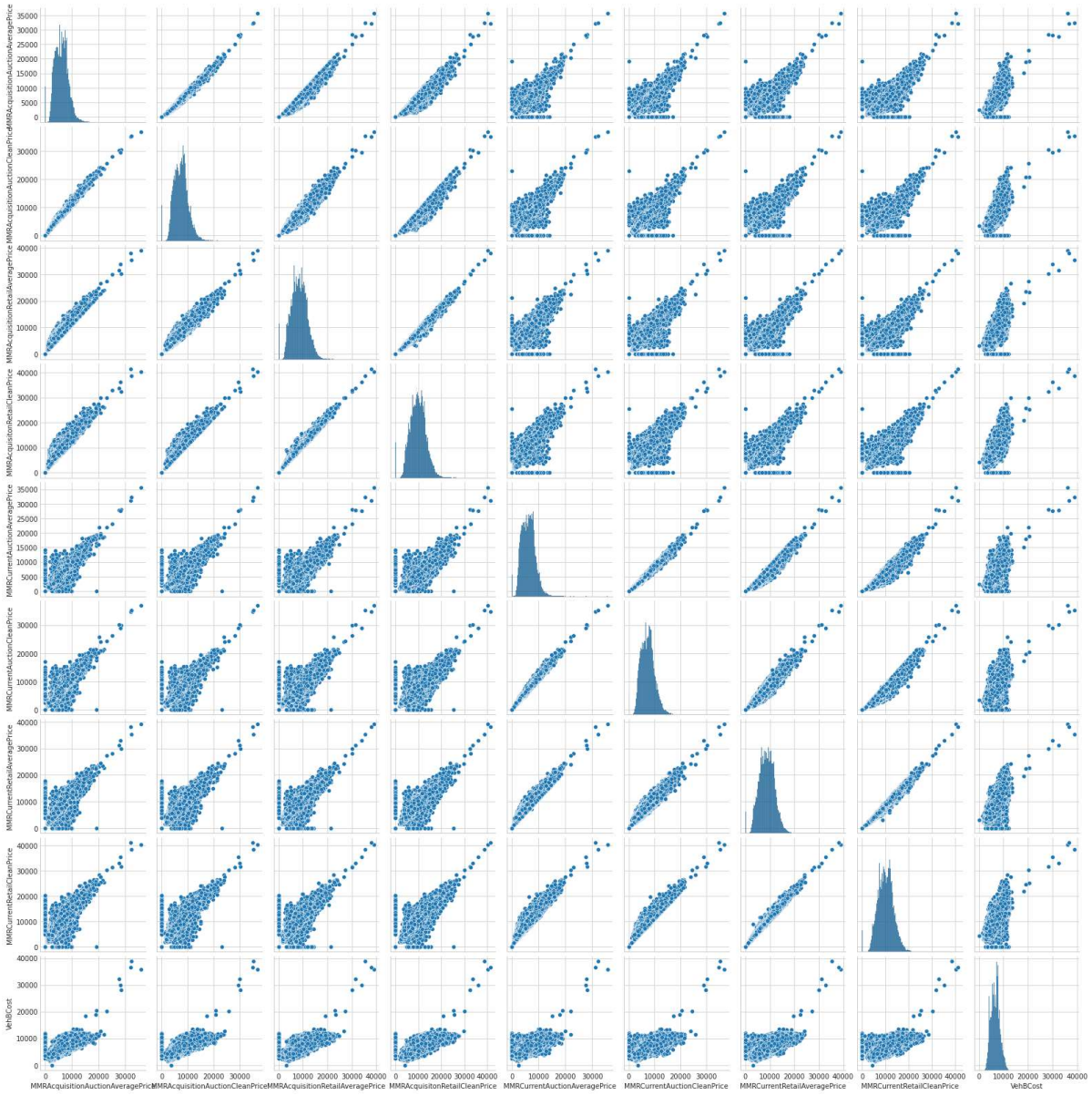
Out[17]:
```
<AxesSubplot:>
```



In [18]:
```
#miteinander stark korreliert!!!
#MMRAcquisitionAuctionAveragePrice, MMRAcquisitionAuctionCleanPrice,
#MMRAcquisitionRetailAveragePrice, MMRAcquisitonRetailCleanPrice,
#MMRCurrentAuctionAveragePrice, MMRCurrentAuctionCleanPrice,
#MMRCurrentRetailAveragePrice, MMRCurrentRetailCleanPrice
```

In [19]:
```
#die Korrelationen als Grafik zu sehen

_ = ['MMRAcquisitionAuctionAveragePrice', 'MMRAcquisitionAuctionCleanPrice', 'MMRAc

sns.pairplot(df[_])
```

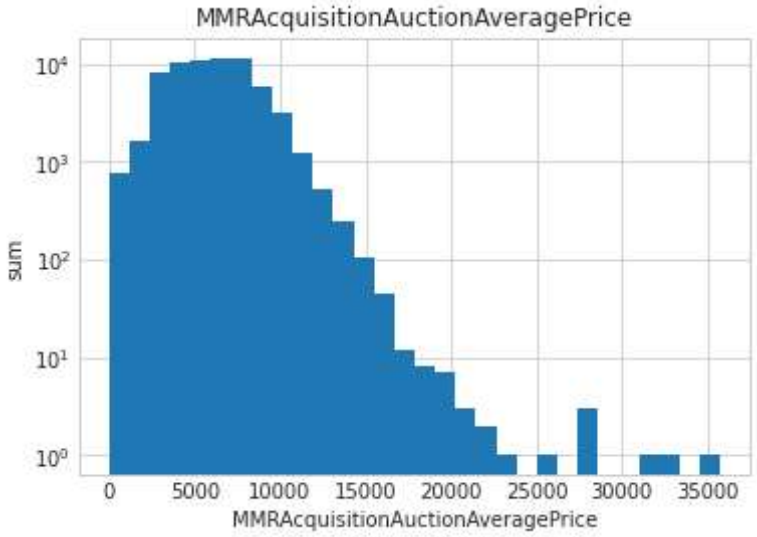
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x7f6e08dacd00>
```

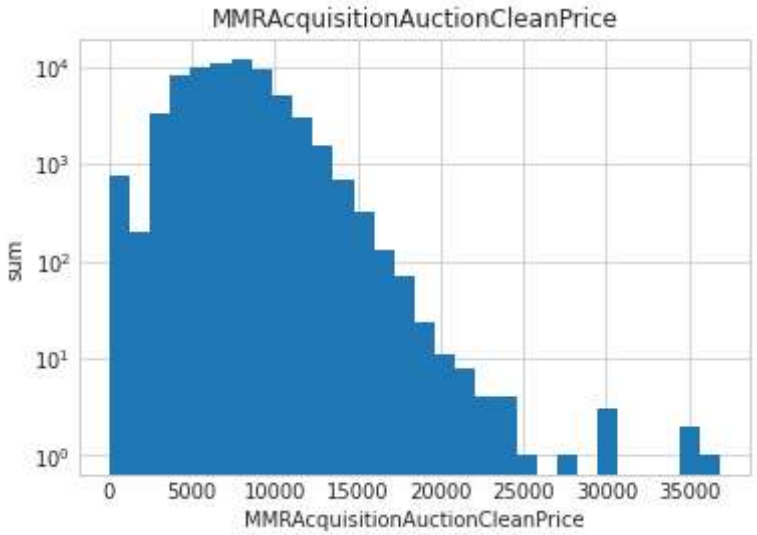


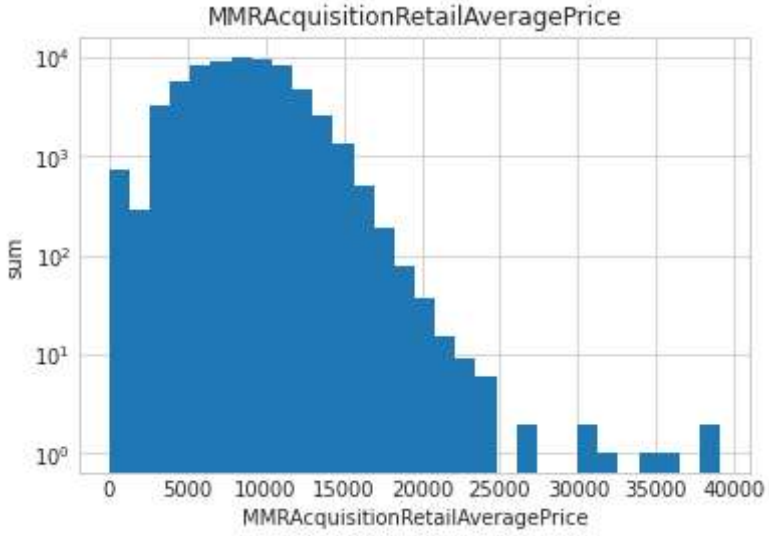
In [ ]:

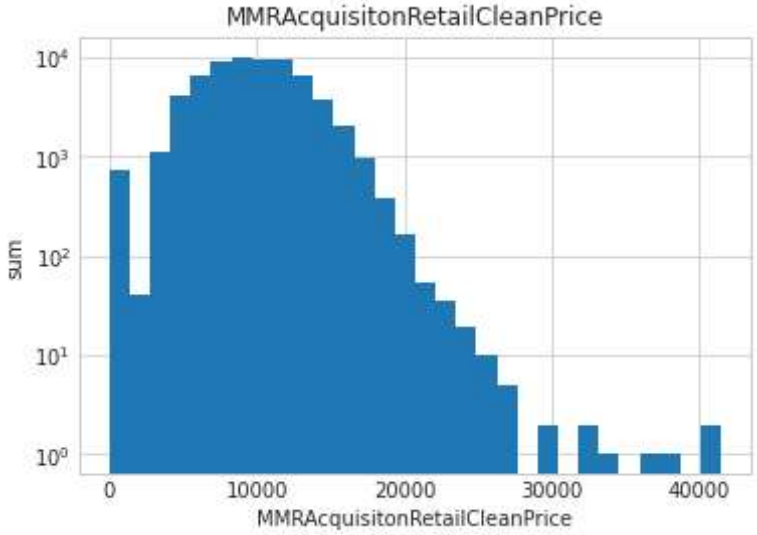
In [20]:
```
def printHistogram(data, columns, logy):
    '''
    Plot histograms for specified columns in the given DataFrame.

    Parameters:
    - data (DataFrame): The input DataFrame containing the data.
    - columns (list): A list of column names for which histograms will be plotted.
    - logy (bool): If True, the y-axis will be displayed in log scale; if False, ir
    '''
    for column in columns:
        fig, ax = plt.subplots()
        data.loc[:, column].plot(legend=False, ax=ax, kind='hist', logy=logy, bins=
        ax.set(xlabel = column, ylabel='sum', title=column)
        plt.show()


liste_for_visualisation= ['MMRAcquisitionAuctionAveragePrice',
                          'MMRAcquisitionAuctionCleanPrice',
                          'MMRAcquisitionRetailAveragePrice',
                          'MMRAcquisitonRetailCleanPrice',
                          'MMRCurrentAuctionAveragePrice',
                          'MMRCurrentAuctionCleanPrice',
                          'MMRCurrentRetailAveragePrice',
```

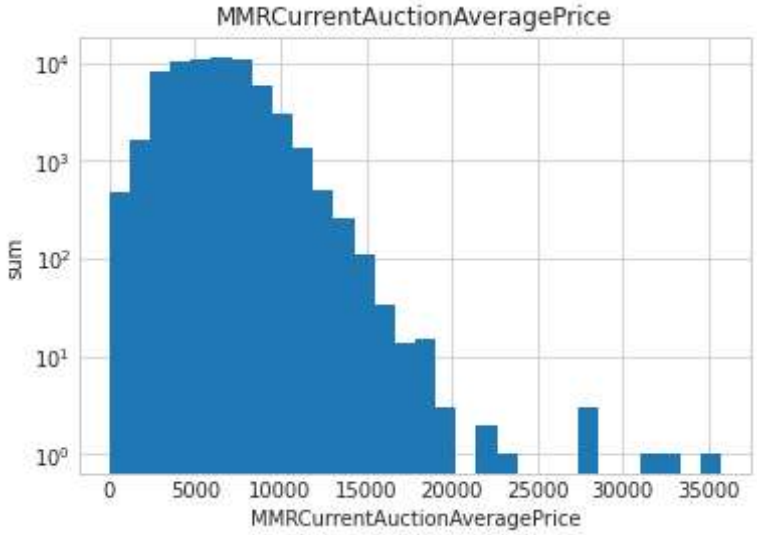
```
                          'MMRCurrentRetailCleanPrice',
                          'VehBCost',
                          'VehOdo',
                          'WarrantyCost']
printHistogram(df, liste_for_visualisation , True)

# logy=True wurde gewählt, da Ausreißer leichter zu beobachten sind.
```

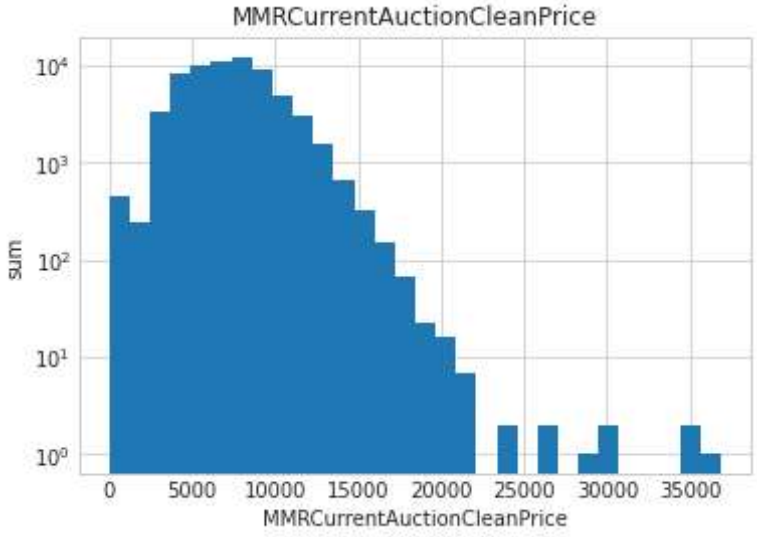


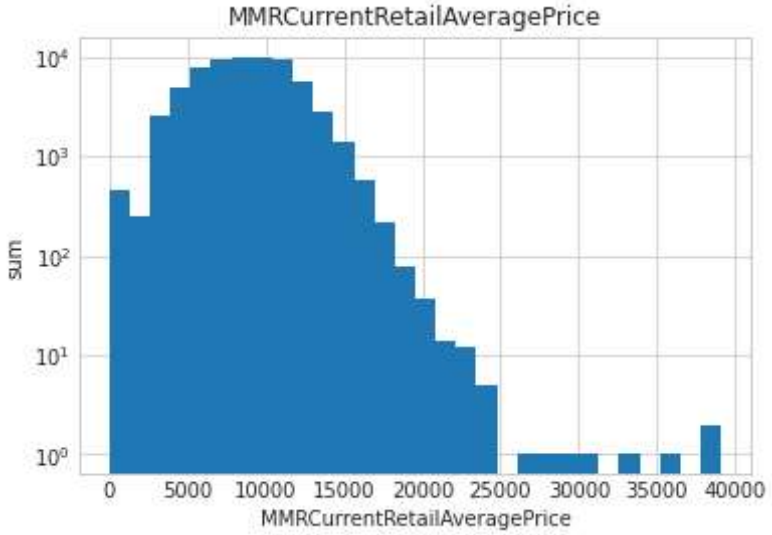


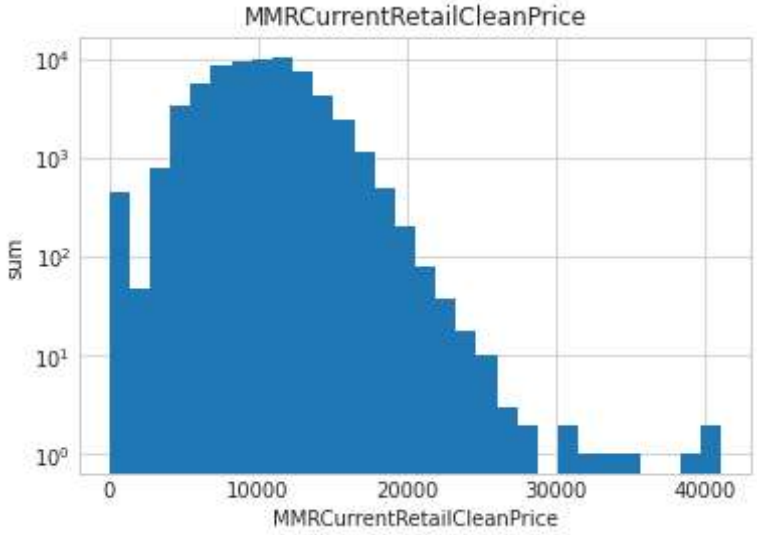


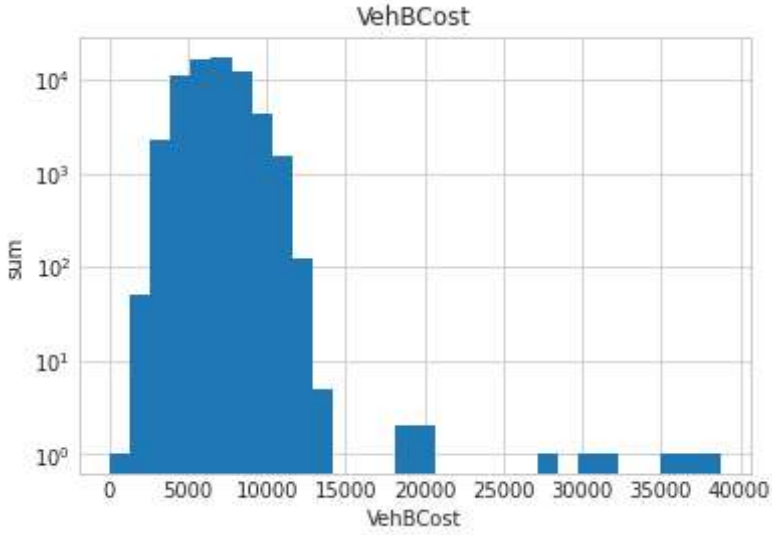

MMRAcquisitonRetailCleanPrice
sum
10^4
10^3
10^2
10^1
10^0
0
10000
20000
30000
40000
MMRAcquisitonRetailCleanPrice

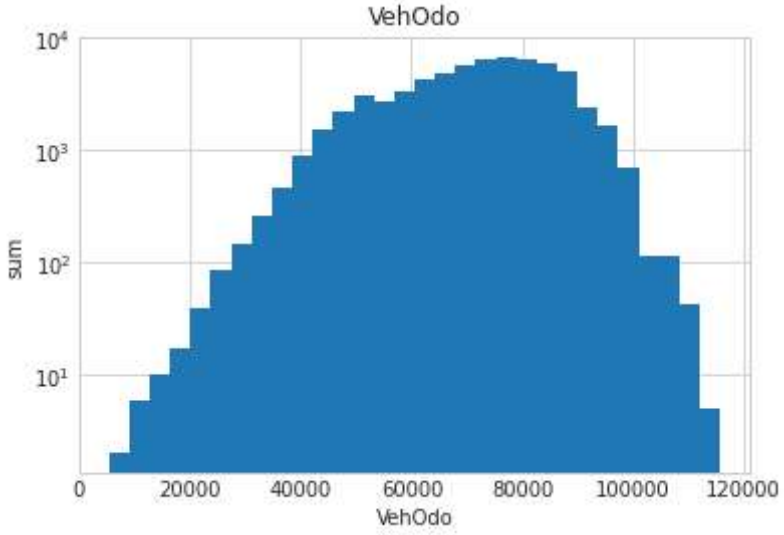

MMRCurrentAuctionAveragePrice
sum
10^4
10^3
10^2
10^1
10^0
0
5000
10000
15000
20000
25000
30000
35000
MMRCurrentAuctionAveragePrice

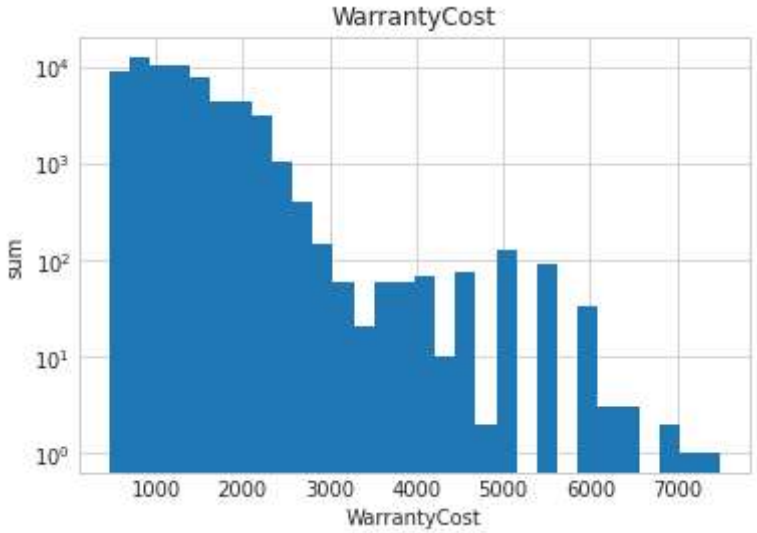

MMRCurrentAuctionCleanPrice
sum
10^4
10^3
10^2
10^1
10^0
0
5000
10000
15000
20000
25000
30000
35000
MMRCurrentAuctionCleanPrice


MMRCurrentRetailAveragePrice
sum
10^4
10^3
10^2
10^1
10^0
0
5000
10000
15000
20000
25000
30000
35000
40000
MMRCurrentRetailAveragePrice


MMRCurrentRetailCleanPrice
sum
10^4
10^3
10^2
10^1
10^0
0
10000
20000
30000
40000
MMRCurrentRetailCleanPrice

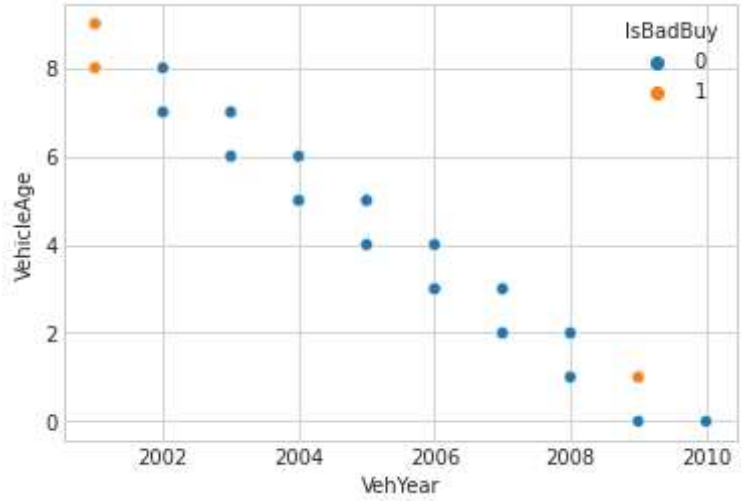

VehBCost
sum
10^4
10^3
10^2
10^1
10^0
0
5000
10000
15000
20000
25000
30000
35000
40000
VehBCost





In [21]:
```
#Einige Visualisierung um die Daten besser zu verstehen
```

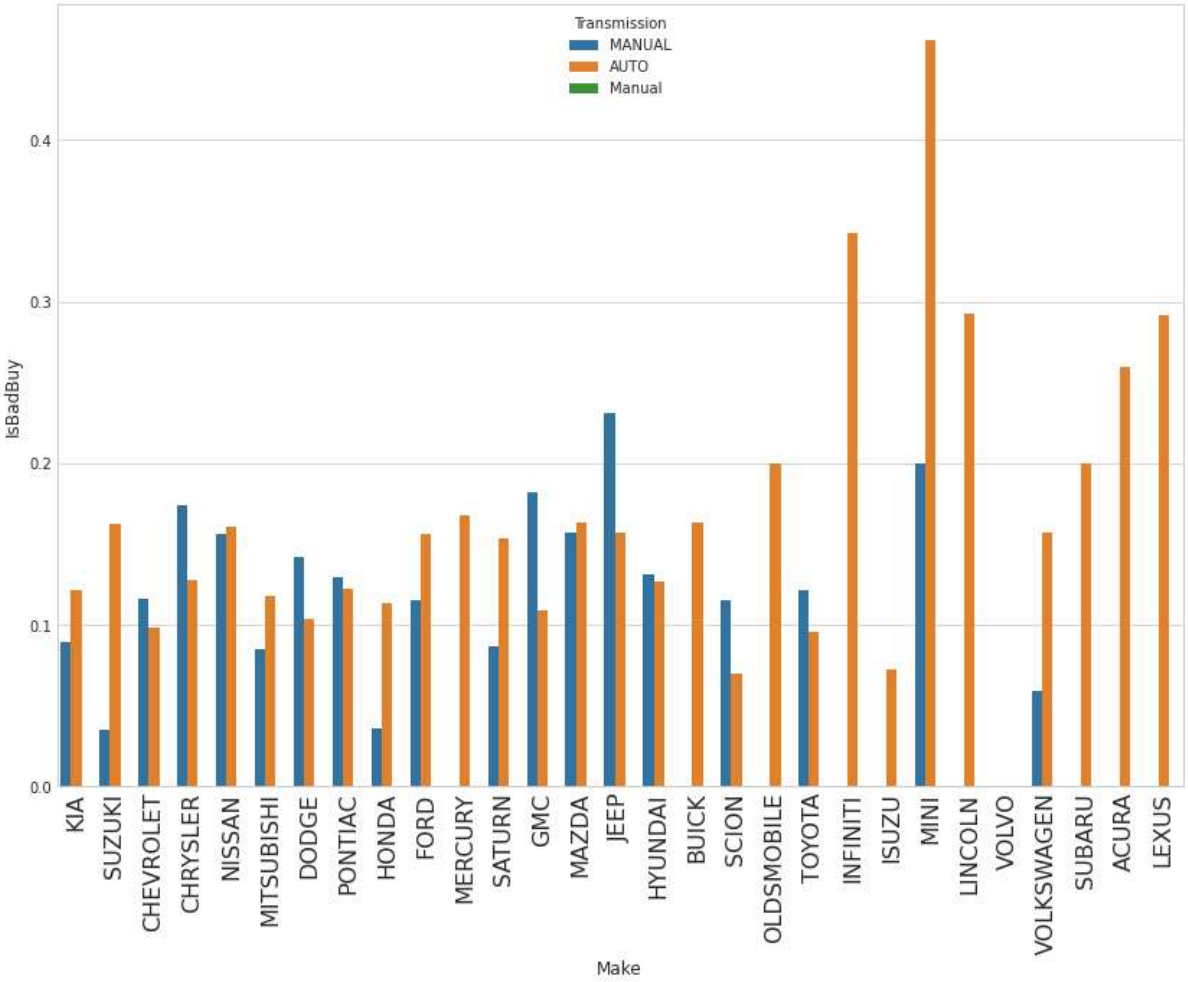
In [22]:
```
#Vehyear/VehicleAge-->miteinander negativ korreliert

sns.scatterplot(x="VehYear", y="VehicleAge", hue='IsBadBuy',data=df)
plt.show()
```



In [23]:
```
grouped_mean = df.groupby('Make')['IsBadBuy'].mean()
grouped_mean
```

Out[23]:
```
Make
ACURA          0.250000
BUICK          0.163110
CHEVROLET      0.098132
CHRYSLER       0.128756
DODGE          0.103647
FORD           0.154028
GMC            0.110727
HONDA          0.108647
HYUNDAI        0.127103
INFINITI       0.342105
ISUZU          0.072000
JEEP           0.158429
KIA            0.118145
LEXUS          0.291667
LINCOLN        0.292135
MAZDA          0.163080
MERCURY        0.167901
MINI           0.347826
MITSUBISHI     0.116052
NISSAN         0.160215
OLDSMOBILE     0.200000
PONTIAC        0.122679
SATURN         0.143801
SCION          0.080357
SUBARU         0.192308
SUZUKI         0.150502
TOYOTA         0.097250
VOLKSWAGEN     0.142857
VOLVO          0.000000
Name: IsBadBuy, dtype: float64
```

In [24]:
```
plt.figure(figsize=(14, 10))
ax = sns.barplot(x="Make", y="IsBadBuy",hue='Transmission', data=df, ci= None)
ax.set_xticklabels(ax.get_xticklabels(),fontsize=16, rotation=90)
ax.set_xlabel("Make", fontsize='large')
ax.set_ylabel("IsBadBuy", fontsize='large')
plt.show()
```



In [25]:
```
categorical_columns
```

Out[25]:
```
['Auction',
 'Make',
 'Model',
 'Trim',
 'SubModel',
 'Color',
 'Transmission',
 'WheelType',
 'Nationality',
 'Size',
 'TopThreeAmericanName',
 'PRIMEUNIT',
 'AUCGUART',
 'VNST']
```

In [26]:
```
def groupby_target(column):
    # Relationship between target and categorical columns
    grouped_data = df[[column, "IsBadBuy"]].groupby(column).mean().sort_values(by='
    value_counts = df[column].value_counts()

    # Add value counts to the grouped data
    grouped_data['Value Counts'] = value_counts

    # Print the updated grouped data
    print(grouped_data)

    print(50*'--')
```

In [27]:
```
cols= ['Auction',
       'VehYear',
```

```
        'VehicleAge',
        'Make',
        'Color',
        'Transmission',
        'WheelType',
        'WheelTypeID',
        'Nationality',
        'Size',
        'TopThreeAmericanName',
        'BYRNO',
        'VNST']
for i in cols:
    groupby_target(i)
```

```
         IsBadBuy  Value Counts
Auction
ADESA    0.151787         13038
OTHER    0.119426         15675
MANHEIM  0.115236         36907
--------------------------------------------------------------------------------
-----------------
         IsBadBuy  Value Counts
VehYear
2001     0.286798          1318
2002     0.244415          3044
2003     0.194405          5612
2004     0.154931          9146
2005     0.132070         13985
2006     0.094610         15305
2007     0.078483         10257
2008     0.055761          6205
2009     0.037483           747
2010     0.000000             1
--------------------------------------------------------------------------------
-----------------
            IsBadBuy  Value Counts
VehicleAge
9           0.305115           567
8           0.275151          1988
7           0.215260          4181
6           0.181260          7161
5           0.146175         11712
4           0.110128         15364
3           0.082841         14208
2           0.064141          7655
1           0.040978          2782
0           0.000000             2
--------------------------------------------------------------------------------
-----------------
            IsBadBuy  Value Counts
Make
MINI        0.347826            23
INFINITI    0.342105            38
LINCOLN     0.292135            89
LEXUS       0.291667            24
ACURA       0.250000            28
OLDSMOBILE  0.200000           215
SUBARU      0.192308            26
MERCURY     0.167901           810
BUICK       0.163110           656
MAZDA       0.163080           883
NISSAN      0.160215          1860
JEEP        0.158429          1477
FORD        0.154028         10167
SUZUKI      0.150502          1196
SATURN      0.143801          1968
VOLKSWAGEN  0.142857           119
CHRYSLER    0.128756          7953
HYUNDAI     0.127103          1605
PONTIAC     0.122679          3823
KIA         0.118145          2243
MITSUBISHI  0.116052           922
GMC         0.110727           578
HONDA       0.108647           451
DODGE       0.103647         11597
CHEVROLET   0.098132         15581
TOYOTA      0.097250          1018
SCION       0.080357           112
```

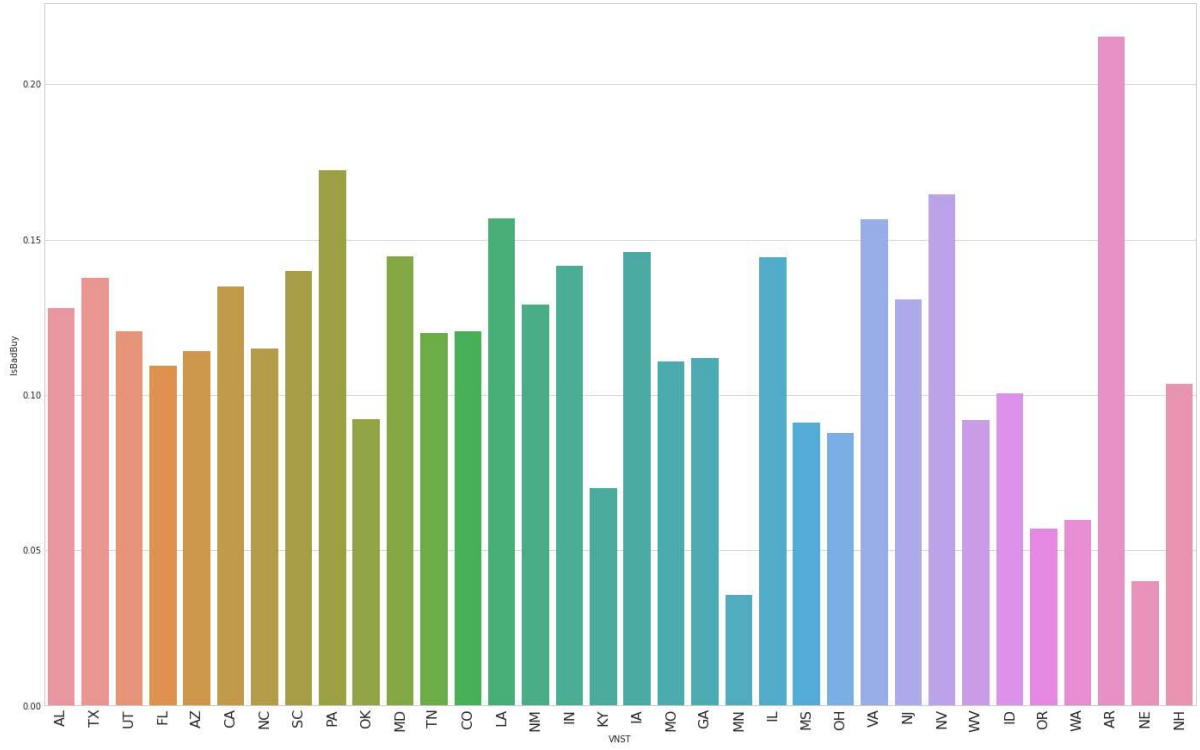
```
ISUZU       0.072000           125
VOLVO       0.000000            33
-------------------------------------------------------------------------------
-----------------
           IsBadBuy  Value Counts
Color
NOT AVAIL  0.261905            84
PURPLE     0.159159           333
BROWN      0.141361           382
GOLD       0.141050          4743
YELLOW     0.140187           214
BEIGE      0.136681          1434
RED        0.133594          5629
OTHER      0.133333           210
MAROON     0.130599          1853
GREEN      0.125874          2860
WHITE      0.125195         10871
SILVER     0.123669         13431
BLACK      0.114236          6828
GREY       0.113894          7068
BLUE       0.113849          9293
ORANGE     0.084211           380
-------------------------------------------------------------------------------
-----------------
              IsBadBuy  Value Counts
Transmission
AUTO          0.123783         63288
MANUAL        0.115799          2323
Manual        0.000000             1
-------------------------------------------------------------------------------
-----------------
           IsBadBuy  Value Counts
WheelType
Special    0.125561           669
Alloy      0.111077         32374
Covers     0.080842         29700
-------------------------------------------------------------------------------
-----------------
             IsBadBuy  Value Counts
WheelTypeID
0.0          0.250000             4
3.0          0.125561           669
1.0          0.111077         32374
2.0          0.080842         29700
-------------------------------------------------------------------------------
-----------------
                IsBadBuy  Value Counts
Nationality
OTHER           0.142857           175
TOP LINE ASIAN  0.134135          3325
OTHER ASIAN     0.131853          7205
AMERICAN        0.121706         54911
-------------------------------------------------------------------------------
-----------------
             IsBadBuy  Value Counts
Size
SPORTS       0.179259           675
COMPACT      0.160475          6487
LARGE SUV    0.159375          1280
SMALL TRUCK  0.147563           759
MEDIUM SUV   0.146798          7323
SMALL SUV    0.138032          2043
VAN          0.126957          5238
MEDIUM       0.116138         27717
```

```
LARGE TRUCK  0.112892          2870
CROSSOVER    0.105196          1578
LARGE        0.093330          7961
SPECIALTY    0.090208          1685
--------------------------------------------------------------------------------
------------------
                     IsBadBuy  Value Counts
TopThreeAmericanName
FORD                 0.156154         11066
OTHER                0.132742         10705
CHRYSLER             0.117004         21025
GM                   0.109334         22820
--------------------------------------------------------------------------------
------------------
       IsBadBuy  Value Counts
BYRNO
99760  1.000000             1
99741  1.000000             1
1041   0.333333             6
1152   0.250000             4
1045   0.220339            59
53245  0.214286            70
10310  0.200000            40
21047  0.189394           132
20833  0.182531          1786
18091  0.181818            11
1085   0.177778            45
16926  0.177162          1191
23359  0.167377          1876
52644  0.158333           360
20740  0.153560          2149
18880  0.153513          3231
20392  0.153509           228
16044  0.151620          1451
1231   0.151429           350
52646  0.146875           320
17212  0.143298          2268
17675  0.140971          2348
52492  0.140496           605
18822  0.139916          1894
18111  0.139571          1913
8655   0.138741          1874
22808  0.137195          1640
21053  0.137116          2538
22916  0.135745          2571
19662  0.128634          1376
52117  0.128609          1143
19064  0.128378           740
25100  0.127235          1454
1156   0.125000             8
11410  0.125000            32
10430  0.122807           171
99740  0.120521           307
835    0.120149          2680
21973  0.119048          2268
23657  0.117834           314
20207  0.117696          1198
19619  0.114007          2456
1191   0.113636            88
3453   0.112848          2623
8172   0.112121           990
1121   0.111111            18
20234  0.110515          1864
19638  0.105139          1693
```

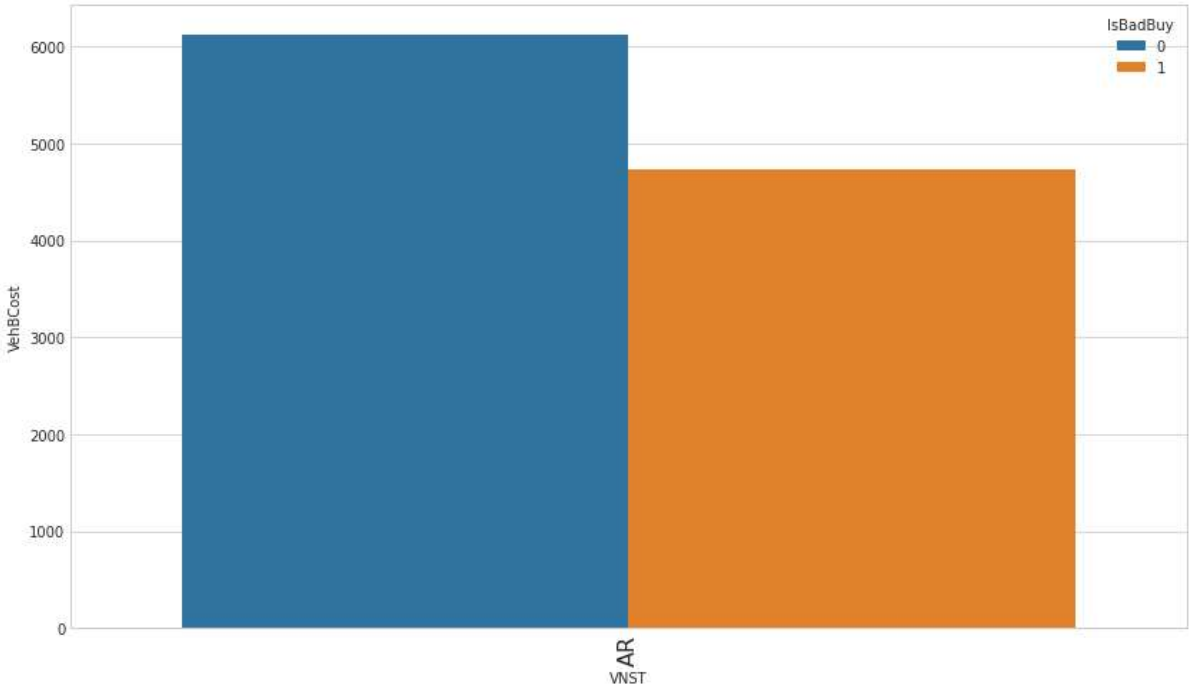
```
20928  0.102015          2333
1035   0.100000            10
52598  0.098672           527
5546   0.096211          2006
1235   0.095794           428
1081   0.093750            32
18881  0.089987          1578
10315  0.088608           158
1051   0.078125            64
1031   0.071429            28
99761  0.056556          3554
1151   0.032787            61
99750  0.029326          2387
1141   0.000000            11
1125   0.000000            10
16369  0.000000             7
10410  0.000000            10
1086   0.000000             1
1082   0.000000             5
10420  0.000000             4
1055   0.000000             9
10510  0.000000             4
11210  0.000000             8
--------------------------------------------------------------------------------
-----------------
      IsBadBuy  Value Counts
VNST
AR    0.215385            65
PA    0.172324           766
NV    0.164683           504
LA    0.156740           319
VA    0.156440          1483
IA    0.146067           445
MD    0.144660          1030
IL    0.144231           416
IN    0.141573           445
SC    0.139942          3823
TX    0.137771         12245
CA    0.134835          6393
NJ    0.130872           298
NM    0.129032           217
AL    0.128000           625
CO    0.120605          4494
UT    0.120558           788
TN    0.119874          1585
NC    0.114857          6347
AZ    0.114015          5587
GA    0.111765          2210
MO    0.110783           677
FL    0.109382          9380
NH    0.103448            87
ID    0.100559           179
OK    0.092188          3200
WV    0.091954           261
MS    0.091121           428
OH    0.087744           718
KY    0.070093           214
WA    0.059829           117
OR    0.056995           193
NE    0.040000            25
MN    0.035714            56
--------------------------------------------------------------------------------
-----------------
```

In [28]:
```
# Die durchschnittliche Anzahl des Montagsautos, die bei der Auktion namens ADESA g

# Mit zunehmendem Alter des Autos steigt das Risiko, zum Montagsauto zu werden.

# Die gekauften Wagen der Marke VOLVO gab es keine 'Montagsautos'.

# Bei gekauften Wagen, deren Farbe nicht angegeben war, ist das Risiko höher.
```

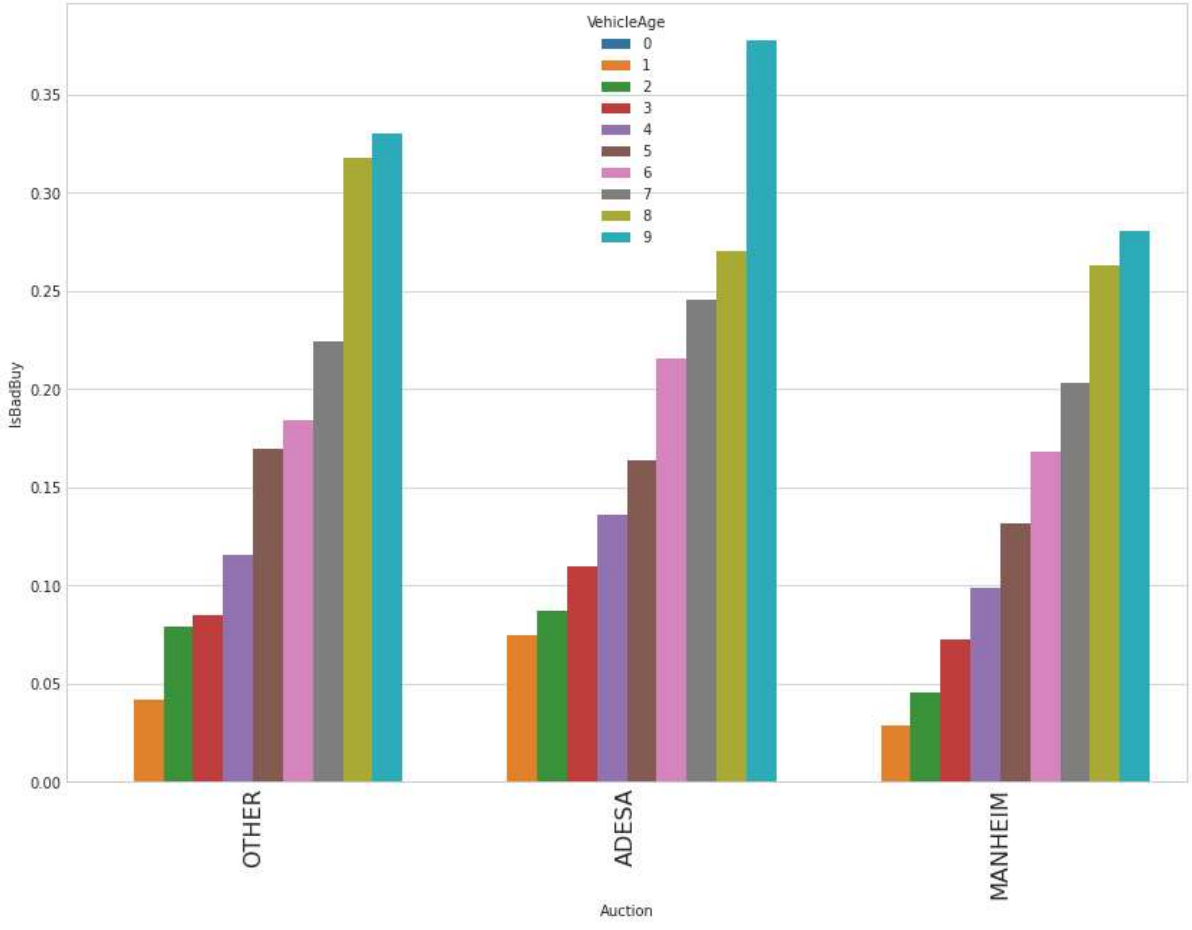
In [29]:
```
#AR: Im Arkansas ist die durchschnittliche Anzahl der Montagsautos höher.

plt.figure(figsize=(24,15))
ax = sns.barplot(x="VNST", y="IsBadBuy",data=df, ci= None)
ax.set_xticklabels(ax.get_xticklabels(),fontsize=16, rotation=90)

plt.show()
```



In [30]:
```
plt.figure(figsize=(14,8))
mask = df[df['VNST']=='AR']['VNST']
ax = sns.barplot(x=mask, y="VehBCost",hue='IsBadBuy', data=df, ci= None)
ax.set_xticklabels(ax.get_xticklabels(),fontsize=16, rotation=90)

plt.show()
```



In [31]:
```
#Mit zunehmendem Alter des Autos steigt das Risiko, zum Montagsauto zu werden.

plt.figure(figsize=(14, 10))
ax = sns.barplot(x="Auction", y="IsBadBuy",hue='VehicleAge', data=df, ci= None)
ax.set_xticklabels(ax.get_xticklabels(),fontsize=16, rotation=90)

plt.show()
```



In [32]:
```
plt.figure(figsize=(24,15))
ax = sns.barplot(x="Make", y="IsBadBuy",hue = 'Auction',data=df, ci= None)
ax.set_xticklabels(ax.get_xticklabels(),fontsize=16, rotation=90)
```