In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn as skl
import matplotlib.pylab as plt
```

executed in 40.9s, finished 23:58:04 2019-05-08

In [2]:

```
train = pd.read_csv('train_technidus.csv')
test = pd.read_csv('test_technidus.csv')
```

executed in 924ms, finished 23:58:05 2019-05-08

1 Data Pre-Processing

In [3]:

```
train.info()
```

executed in 111ms, finished 23:58:05 2019-05-08

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7654 entries, 0 to 7653
Data columns (total 25 columns):
CustomerID               7654 non-null int64
Title                    49 non-null object
FirstName                7654 non-null object
MiddleName               4487 non-null object
LastName                 7654 non-null object
Suffix                   0 non-null float64
AddressLine1             7654 non-null object
AddressLine2             123 non-null object
City                     7654 non-null object
StateProvinceName        7654 non-null object
CountryRegionName        7654 non-null object
PostalCode               7654 non-null object
PhoneNumber              7654 non-null object
BirthDate                7654 non-null object
Education                7654 non-null object
Occupation               7654 non-null object
Gender                   7654 non-null object
MaritalStatus            7654 non-null object
HomeOwnerFlag            7654 non-null int64
NumberCarsOwned          7654 non-null int64
NumberChildrenAtHome     7654 non-null int64
TotalChildren            7654 non-null int64
YearlyIncome             7654 non-null int64
AveMonthSpend            7654 non-null int64
BikeBuyer                7654 non-null int64
dtypes: float64(1), int64(8), object(16)
memory usage: 1.5+ MB
```

In [4]:

```
test.info()
```

executed in 367ms, finished 23:58:05 2019-05-08

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3340 entries, 0 to 3339
Data columns (total 25 columns):
CustomerID              3340 non-null int64
Title                   15 non-null object
FirstName               3340 non-null object
MiddleName              1883 non-null object
LastName                3340 non-null object
Suffix                  1 non-null object
AddressLine1            3340 non-null object
AddressLine2            60 non-null object
City                    3340 non-null object
StateProvinceName       3340 non-null object
CountryRegionName       3340 non-null object
PostalCode              3340 non-null object
PhoneNumber             3340 non-null object
BirthDate               3340 non-null object
Education               3340 non-null object
Occupation              3340 non-null object
Gender                  3340 non-null object
MaritalStatus           3340 non-null object
HomeOwnerFlag           3340 non-null int64
NumberCarsOwned         3340 non-null int64
NumberChildrenAtHome    3340 non-null int64
TotalChildren           3340 non-null int64
YearlyIncome            3340 non-null int64
AveMonthSpend           0 non-null float64
BikeBuyer               3340 non-null int64
dtypes: float64(1), int64(7), object(17)
memory usage: 652.4+ KB
```

In [5]:

```
#Print out all the columns that have not more than 30% null values
nn_cols=[col for col in train.columns if train[col].count()>=0.7*len(train)]
print(nn_cols)
```

executed in 141ms, finished 23:58:06 2019-05-08

```
['CustomerID', 'FirstName', 'LastName', 'AddressLine1', 'City', 'StateProvin
ceName', 'CountryRegionName', 'PostalCode', 'PhoneNumber', 'BirthDate', 'Edu
cation', 'Occupation', 'Gender', 'MaritalStatus', 'HomeOwnerFlag', 'NumberCa
rsOwned', 'NumberChildrenAtHome', 'TotalChildren', 'YearlyIncome', 'AveMonth
Spend', 'BikeBuyer']
```

In [6]:

```
train=train[nn_cols]
test=test[nn_cols]
```

executed in 110ms, finished 23:58:06 2019-05-08

In [7]:

```
1 train.isnull().sum()
```

executed in 135ms, finished 23:58:06 2019-05-08

Out[7]:

```
CustomerID              0
FirstName               0
LastName                0
AddressLine1            0
City                    0
StateProvinceName       0
CountryRegionName       0
PostalCode              0
PhoneNumber             0
BirthDate               0
Education               0
Occupation              0
Gender                  0
MaritalStatus           0
HomeOwnerFlag           0
NumberCarsOwned         0
NumberChildrenAtHome    0
TotalChildren           0
YearlyIncome            0
AveMonthSpend           0
BikeBuyer               0
dtype: int64
```

In [8]:

```
1 test.isnull().sum()
```

executed in 151ms, finished 23:58:06 2019-05-08

Out[8]:

```
CustomerID                 0
FirstName                  0
LastName                   0
AddressLine1               0
City                       0
StateProvinceName          0
CountryRegionName          0
PostalCode                 0
PhoneNumber                0
BirthDate                  0
Education                  0
Occupation                 0
Gender                     0
MaritalStatus              0
HomeOwnerFlag              0
NumberCarsOwned            0
NumberChildrenAtHome       0
TotalChildren              0
YearlyIncome               0
AveMonthSpend           3340
BikeBuyer                  0
dtype: int64
```

In [9]:

```
1  train.nunique()
```

executed in 212ms, finished 23:58:06 2019-05-08

Out[9]:

```
CustomerID              7654
FirstName                612
LastName                 298
AddressLine1            6583
City                     252
StateProvinceName         46
CountryRegionName          6
PostalCode               304
PhoneNumber             3831
BirthDate               5168
Education                  5
Occupation                 5
Gender                     2
MaritalStatus              2
HomeOwnerFlag              2
NumberCarsOwned            5
NumberChildrenAtHome       6
TotalChildren              6
YearlyIncome            7449
AveMonthSpend            150
BikeBuyer                  2
dtype: int64
```

In [10]:

```
1  test.nunique()
```

executed in 70ms, finished 23:58:06 2019-05-08

Out[10]:

```
CustomerID              3340
FirstName                583
LastName                 251
AddressLine1            3122
City                     237
StateProvinceName         38
CountryRegionName          6
PostalCode               286
PhoneNumber             1773
BirthDate               2766
Education                  5
Occupation                 5
Gender                     2
MaritalStatus              2
HomeOwnerFlag              2
NumberCarsOwned            5
NumberChildrenAtHome       6
TotalChildren              6
YearlyIncome            3306
AveMonthSpend              0
BikeBuyer                  2
dtype: int64
```

In [11]:

```
#Drop features that are unlikely to be informative
to_drop = ['FirstName','LastName','City','StateProvinceName','AddressLine1','PostalCo

train.drop(to_drop,inplace=True,axis=1)
test.drop(to_drop,inplace=True,axis=1)
```

executed in 255ms, finished 23:58:07 2019-05-08

In [12]:

```
#Convert BirthDate to Year,Month
train['BirthYear']=pd.to_datetime(train['BirthDate']).dt.year;
train['BirthMonth']=pd.to_datetime(train['BirthDate']).dt.month;
train.drop(['BirthDate'],axis=1,inplace=True)
```

executed in 8.47s, finished 23:58:15 2019-05-08

In [13]:

```
test['BirthYear']=pd.to_datetime(test['BirthDate']).dt.year;
test['BirthMonth']=pd.to_datetime(test['BirthDate']).dt.month;
test.drop(['BirthDate'],axis=1,inplace=True)
```

executed in 3.63s, finished 23:58:19 2019-05-08

In [14]:

```
train.isnull().sum()
```

executed in 36ms, finished 23:58:19 2019-05-08

Out[14]:

```
CustomerID            0
CountryRegionName     0
Education             0
Occupation            0
Gender                0
MaritalStatus         0
HomeOwnerFlag         0
NumberCarsOwned       0
NumberChildrenAtHome  0
TotalChildren         0
YearlyIncome          0
AveMonthSpend         0
BikeBuyer             0
BirthYear             0
BirthMonth            0
dtype: int64
```

In [15]:

```
test.isnull().sum()
```

executed in 142ms, finished 23:58:19 2019-05-08

Out[15]:

```
CustomerID                0
CountryRegionName         0
Education                 0
Occupation                0
Gender                    0
MaritalStatus             0
HomeOwnerFlag             0
NumberCarsOwned           0
NumberChildrenAtHome      0
TotalChildren             0
YearlyIncome              0
AveMonthSpend          3340
BikeBuyer                 0
BirthYear                 0
BirthMonth                0
dtype: int64
```

In [16]:

```
train.nunique()
```

executed in 213ms, finished 23:58:19 2019-05-08

Out[16]:

```
CustomerID             7654
CountryRegionName         6
Education                 5
Occupation                5
Gender                    2
MaritalStatus             2
HomeOwnerFlag             2
NumberCarsOwned           5
NumberChildrenAtHome      6
TotalChildren             6
YearlyIncome           7449
AveMonthSpend           150
BikeBuyer                 2
BirthYear                63
BirthMonth               12
dtype: int64
```

In [17]:

```
1   test.nunique()
```

executed in 165ms, finished 23:58:19 2019-05-08

Out[17]:

```
CustomerID              3340
CountryRegionName          6
Education                  5
Occupation                 5
Gender                     2
MaritalStatus              2
HomeOwnerFlag              2
NumberCarsOwned            5
NumberChildrenAtHome       6
TotalChildren              6
YearlyIncome            3306
AveMonthSpend              0
BikeBuyer                  2
BirthYear                 54
BirthMonth                12
dtype: int64
```

In [18]:

```
1   train.describe()
```

executed in 432ms, finished 23:58:20 2019-05-08

Out[18]:

	CustomerID	HomeOwnerFlag	NumberCarsOwned	NumberChildrenAtHome	TotalChildre
count	7654.000000	7654.000000	7654.000000	7654.000000	7654.00000
mean	18784.735824	0.695192	1.581657	1.253201	2.16109
std	4795.026146	0.460356	1.186209	1.659555	1.73142
min	11001.000000	0.000000	0.000000	0.000000	0.00000
25%	14760.750000	0.000000	1.000000	0.000000	1.00000
50%	18479.500000	1.000000	2.000000	0.000000	2.00000
75%	22431.500000	1.000000	2.000000	2.000000	4.00000
max	29481.000000	1.000000	4.000000	5.000000	5.00000

In [19]:

```
test.describe()
```

executed in 165ms, finished 23:58:20 2019-05-08

Out[19]:

	CustomerID	HomeOwnerFlag	NumberCarsOwned	NumberChildrenAtHome	TotalChildre
count	3340.000000	3340.000000	3340.000000	3340.000000	3340.00000
mean	23441.067964	0.625150	1.545509	1.182036	2.12095
std	5107.867886	0.484157	1.183150	1.632014	1.70399
min	11026.000000	0.000000	0.000000	0.000000	0.00000
25%	20374.250000	0.000000	1.000000	0.000000	1.00000
50%	25180.500000	1.000000	2.000000	0.000000	2.00000
75%	27368.250000	1.000000	2.000000	2.000000	4.00000
max	29480.000000	1.000000	4.000000	5.000000	5.00000

In [20]:

```
cat_col=[col for col in train.columns if train[col].nunique()<7]
num_col=list(set([col for col in train.columns if train[col].nunique()>7])-set(['Cust
```

executed in 56ms, finished 23:58:20 2019-05-08

In [21]:

```
print('Categorical features are:',cat_col)
print('')
print('Numerical features are:',num_col)
```

executed in 106ms, finished 23:58:20 2019-05-08

```
Categorical features are: ['CountryRegionName', 'Education', 'Occupation',
'Gender', 'MaritalStatus', 'HomeOwnerFlag', 'NumberCarsOwned', 'NumberChildr
enAtHome', 'TotalChildren', 'BikeBuyer']

Numerical features are: ['YearlyIncome', 'BirthYear', 'BirthMonth', 'AveMont
hSpend']
```

2 Exploratory Data Analysis

In [22]:

```
#Distribution of customers for each categorical variable
for col in cat_col:
    print(train[col].value_counts())
    print('')
```

executed in 104ms, finished 23:58:20 2019-05-08

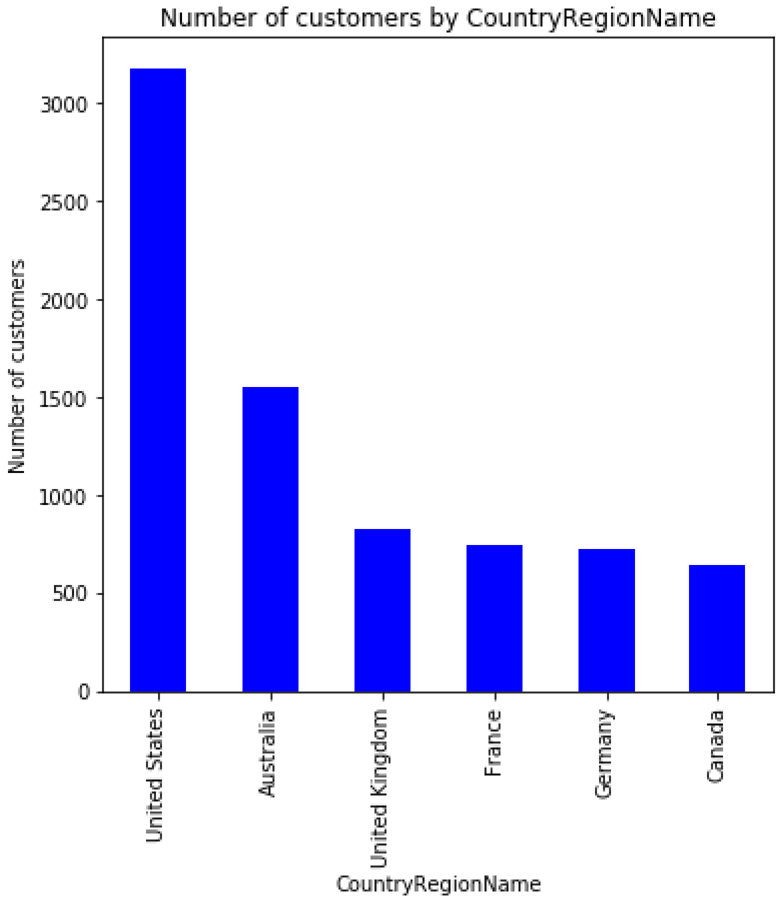
```
United States     3176
Australia         1554
United Kingdom     822
France             745
Germany            717
Canada             640
Name: CountryRegionName, dtype: int64

Bachelors              2337
Partial College        2064
High School            1337
Graduate Degree        1316
Partial High School     600
Name: Education, dtype: int64

Professional      2489
Skilled Manual    1808
Management        1327
Clerical          1077
Manual             953
Name: Occupation, dtype: int64

M    3984
F    3670
Name: Gender, dtype: int64

M    3901
S    3753
Name: MaritalStatus, dtype: int64

1    5321
0    2333
Name: HomeOwnerFlag, dtype: int64

2    2530
1    1973
0    1677
3     823
4     651
Name: NumberCarsOwned, dtype: int64

0    4115
1     936
2     845
3     624
4     576
5     558
Name: NumberChildrenAtHome, dtype: int64

0    1847
2    1338
1    1332
4    1204
```

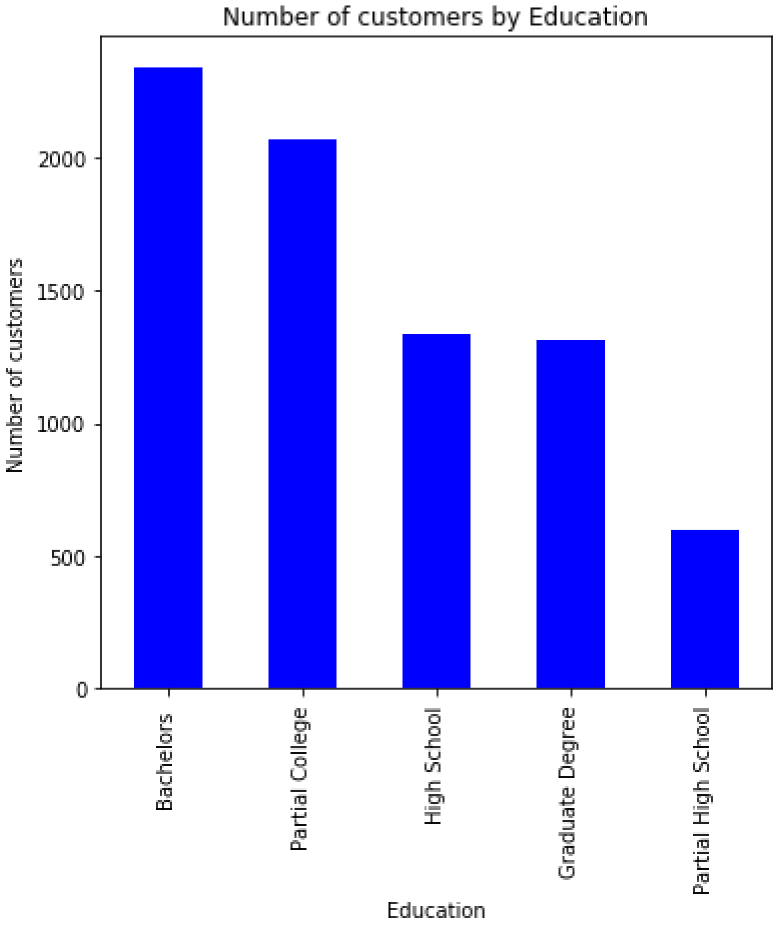
```
3     974
5     959
Name: TotalChildren, dtype: int64

0    3841
1    3813
Name: BikeBuyer, dtype: int64
```

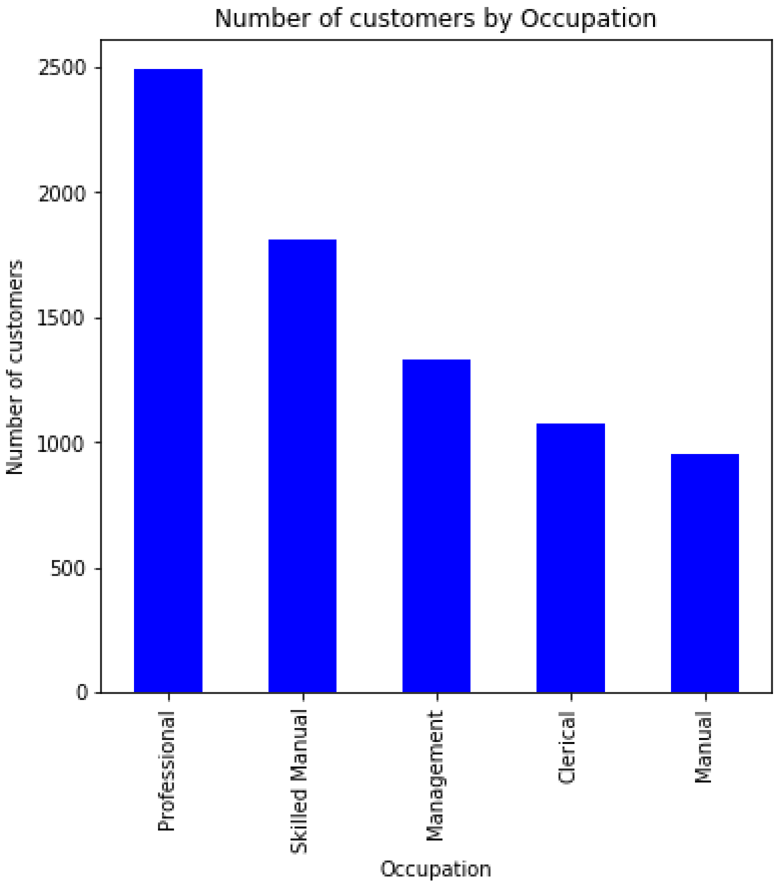
In [23]:

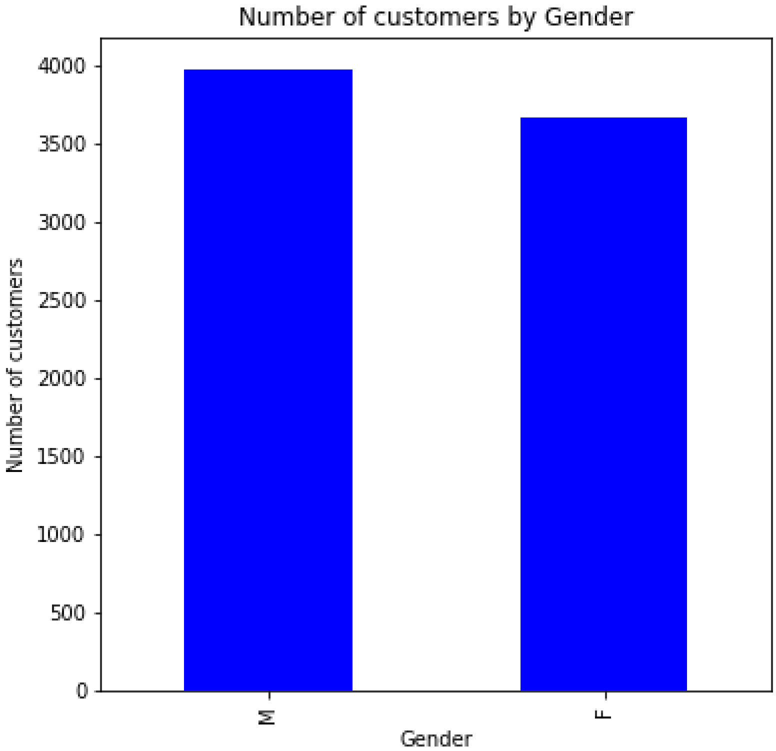
```
for col in cat_col:
        fig = plt.figure(figsize=(6,6))
        ax = fig.gca()
        counts = train[col].value_counts()
        counts.plot.bar(ax = ax, color = 'blue')
        ax.set_title('Number of customers by ' + col)
        ax.set_xlabel(col)
        ax.set_ylabel('Number of customers')
        plt.show()
```

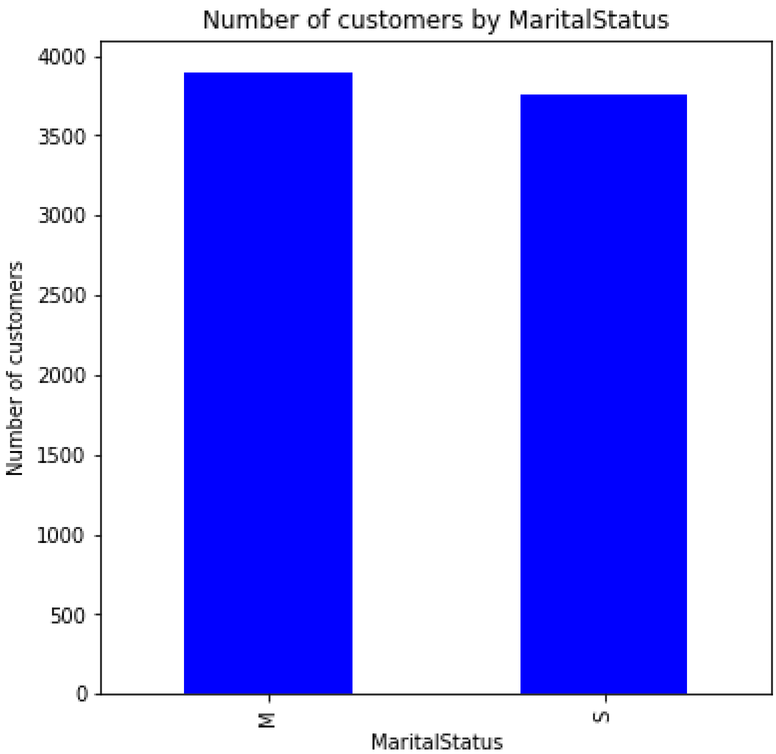
executed in 4.60s, finished 23:58:25 2019-05-08

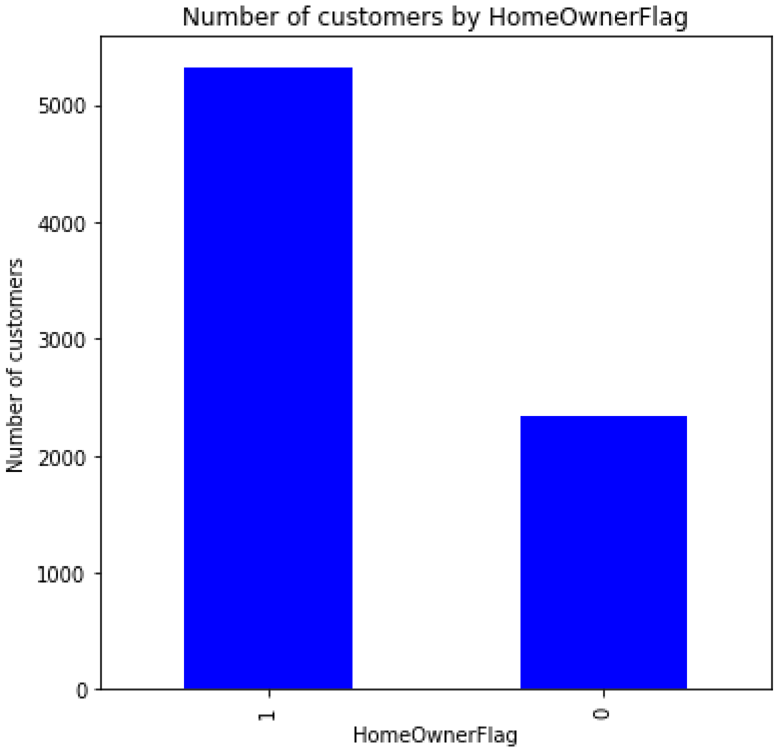


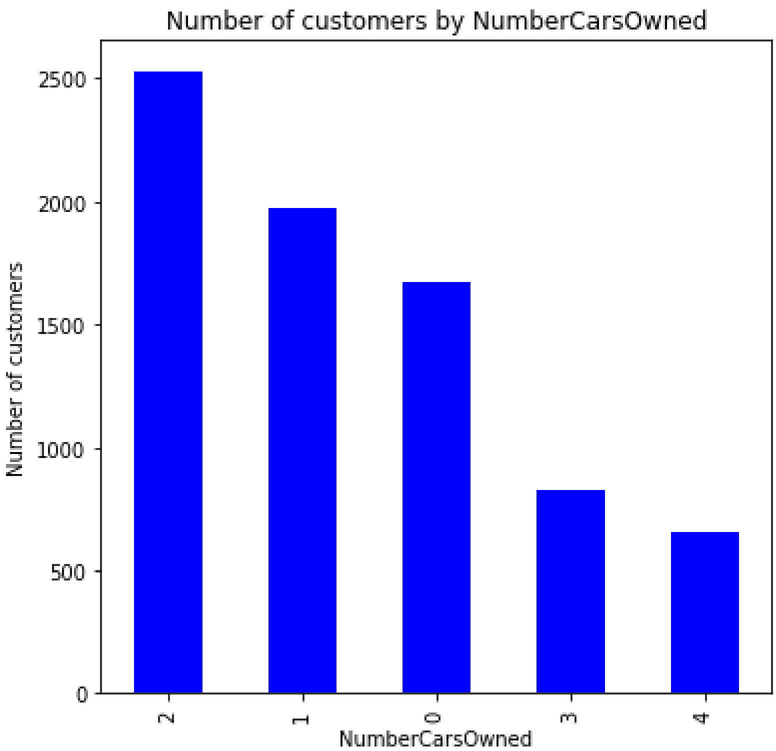

Number of customers by Education
Number of customers
0
500
1000
1500
2000
Bachelors
Partial College
High School
Graduate Degree
Partial High School
Education

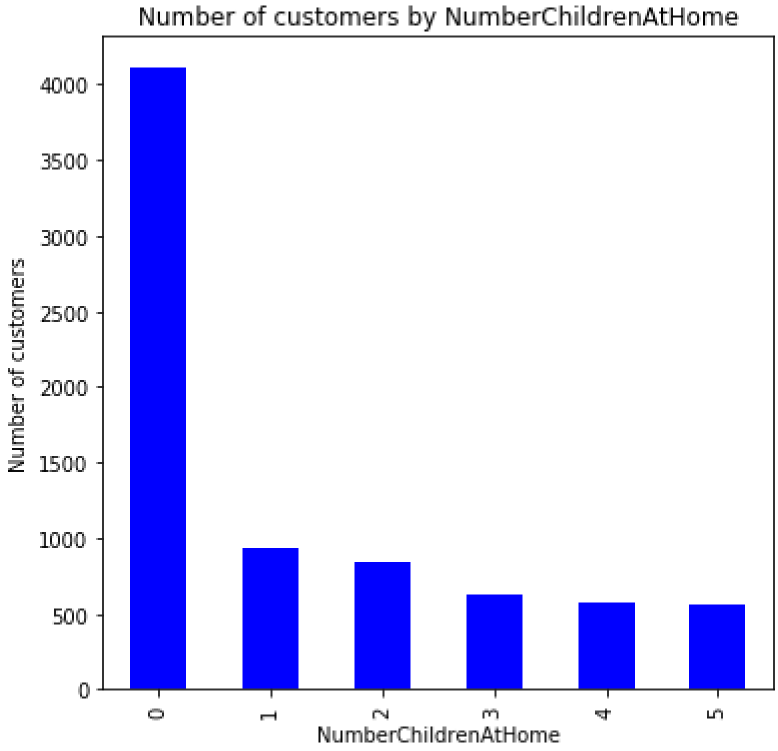

Number of customers by Occupation
Number of customers
0
500
1000
1500
2000
2500
Professional
Skilled Manual
Management
Clerical
Manual
Occupation

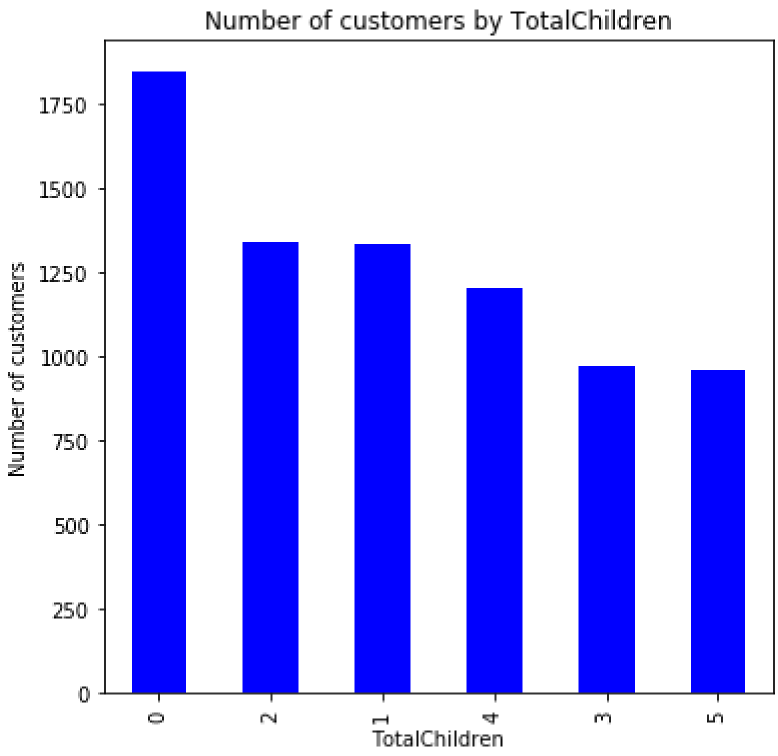

Number of customers by Gender
Number of customers
4000
3500
3000
2500
2000
1500
1000
500
0
M
F
Gender

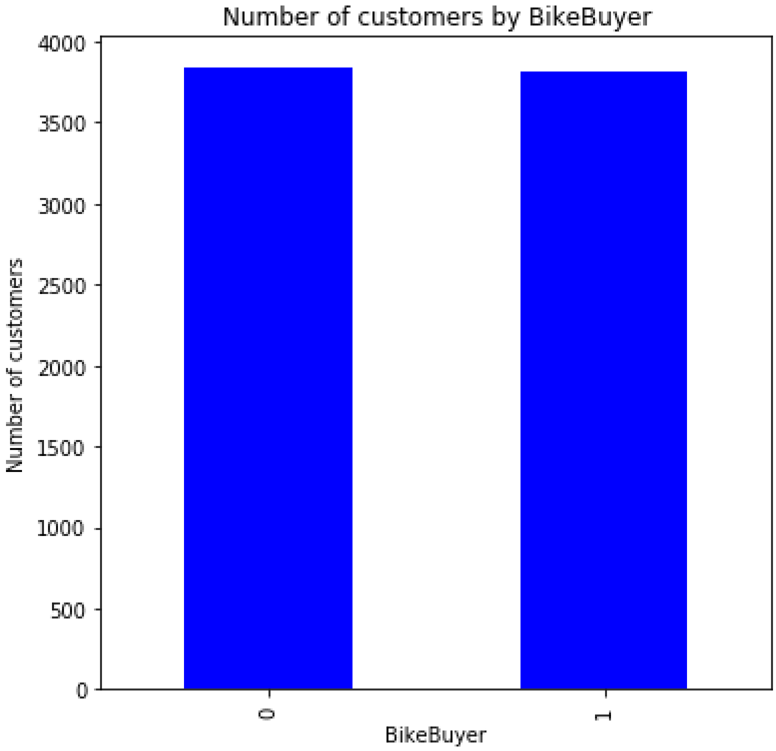

Number of customers by MaritalStatus
Number of customers
4000
3500
3000
2500
2000
1500
1000
500
0
M
S
MaritalStatus


Number of customers by HomeOwnerFlag
Number of customers
0
1000
2000
3000
4000
5000
1
0
HomeOwnerFlag


Number of customers by NumberCarsOwned
Number of customers
0
500
1000
1500
2000
2500
2
1
0
3
4
NumberCarsOwned


Number of customers by NumberChildrenAtHome
Number of customers
4000
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
NumberChildrenAtHome


Number of customers by TotalChildren
Number of customers
1750
1500
1250
1000
750
500
250
0
0
2
1
4
3
5
TotalChildren


Number of customers by BikeBuyer
4000
3500
3000
2500
2000
1500
1000
500
0
Number of customers
0
1
BikeBuyer

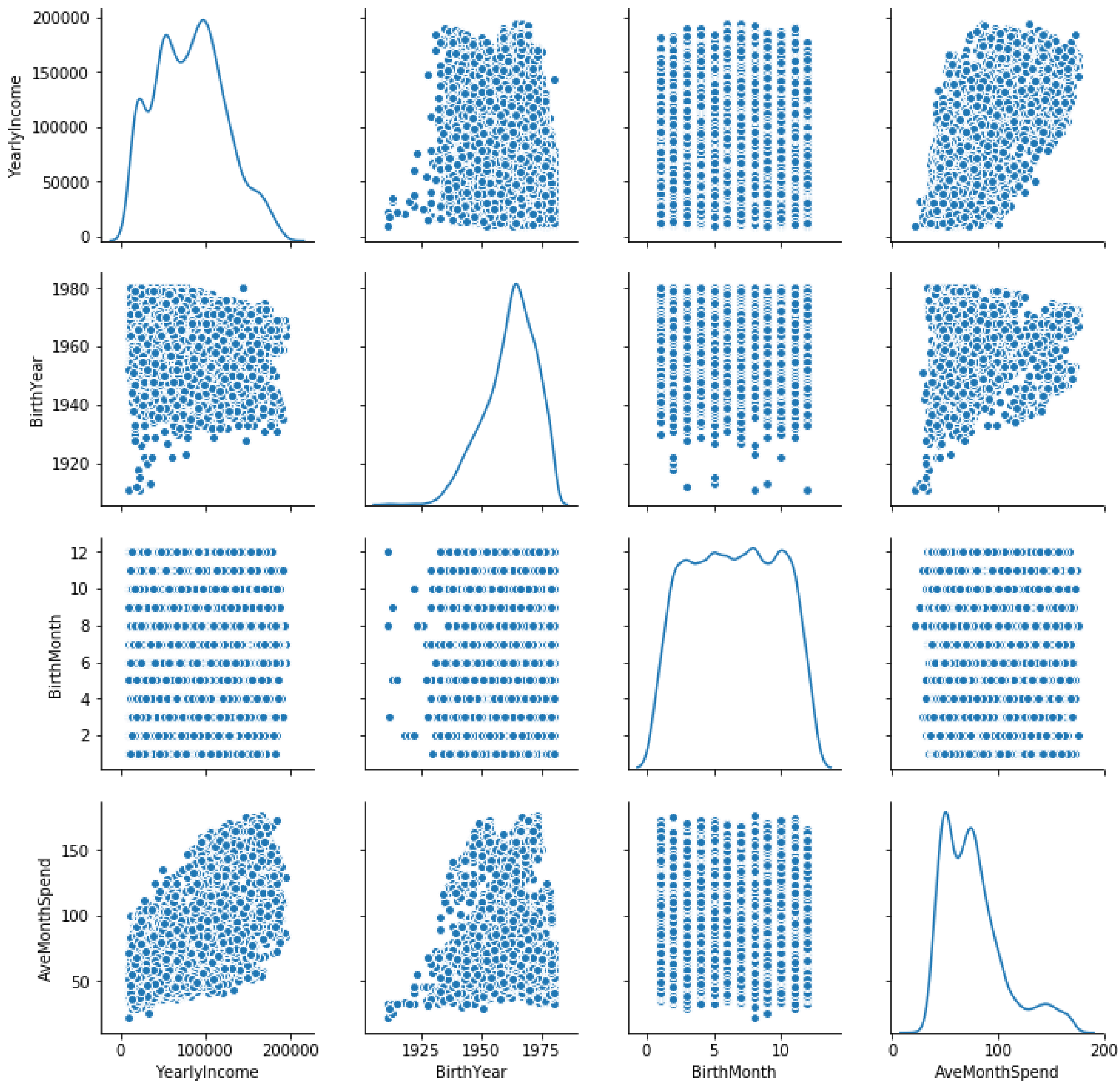
In [24]:

```
sns.pairplot(train[num_col],diag_kind='kde')
```

executed in 8.79s, finished 23:58:34 2019-05-08

Out[24]:

<seaborn.axisgrid.PairGrid at 0x1ce7101d278>



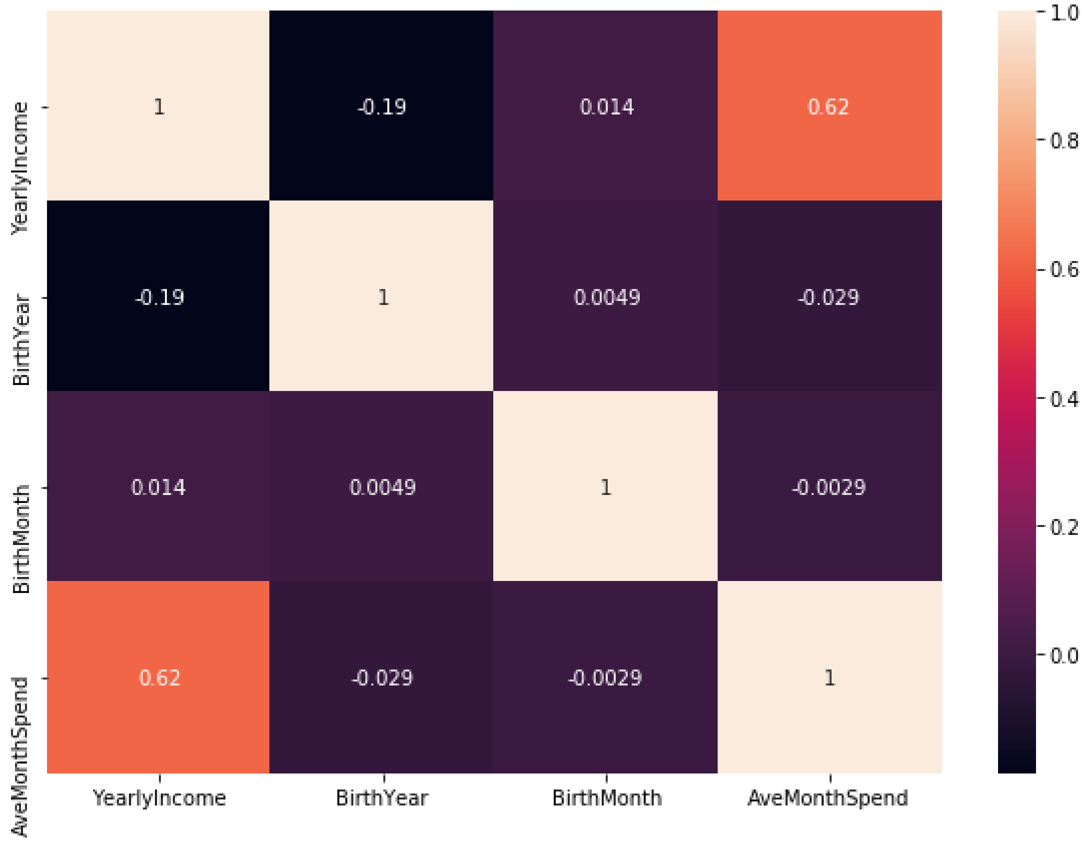
In [25]:

```
plt.figure(figsize=(10,7))
sns.heatmap(train[num_col].corr(),annot=True)
```

executed in 1.27s, finished 23:58:35 2019-05-08

Out[25]:

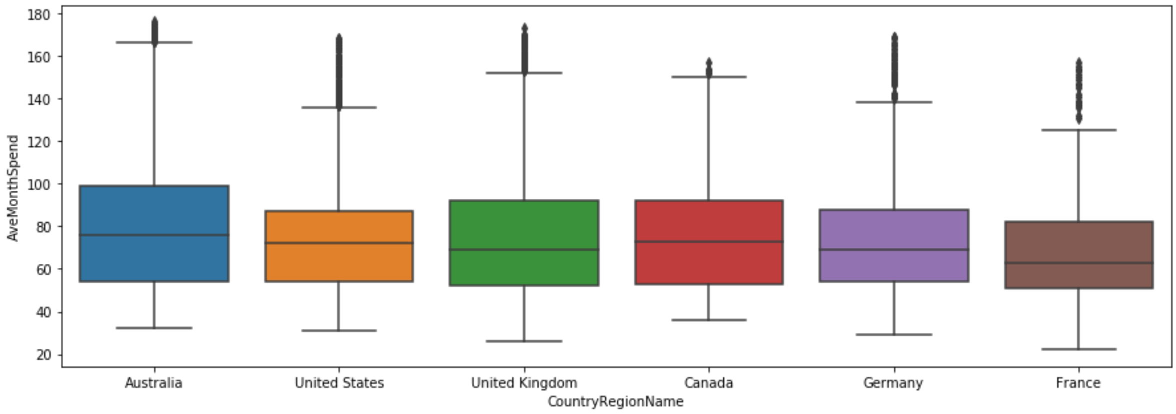
<matplotlib.axes._subplots.AxesSubplot at 0x1ce71447b38>

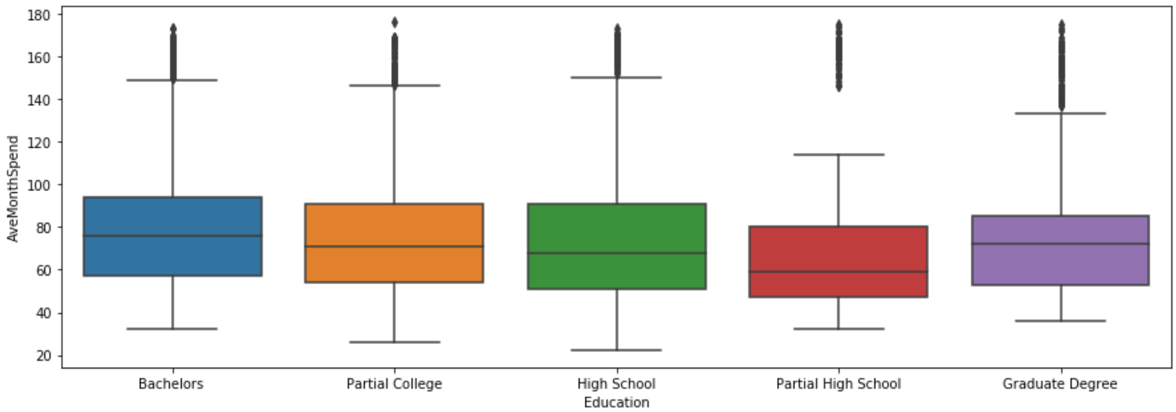


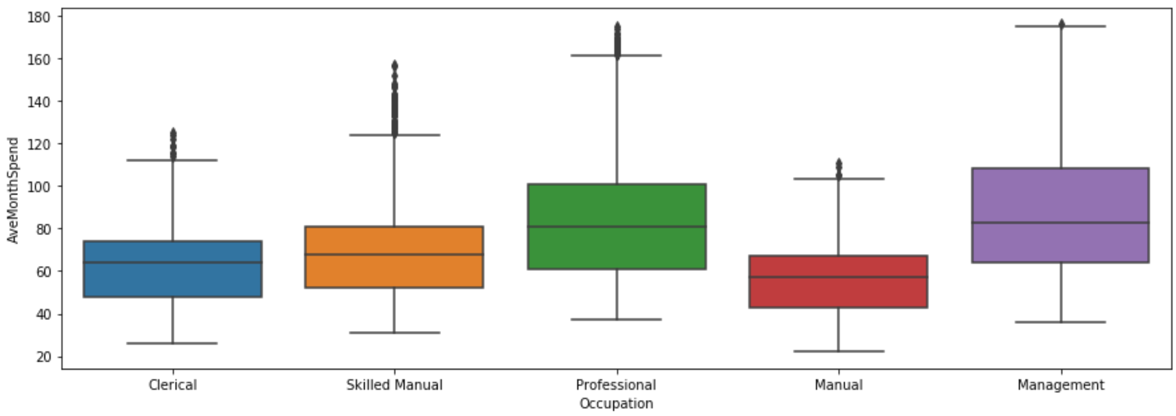
In [26]:

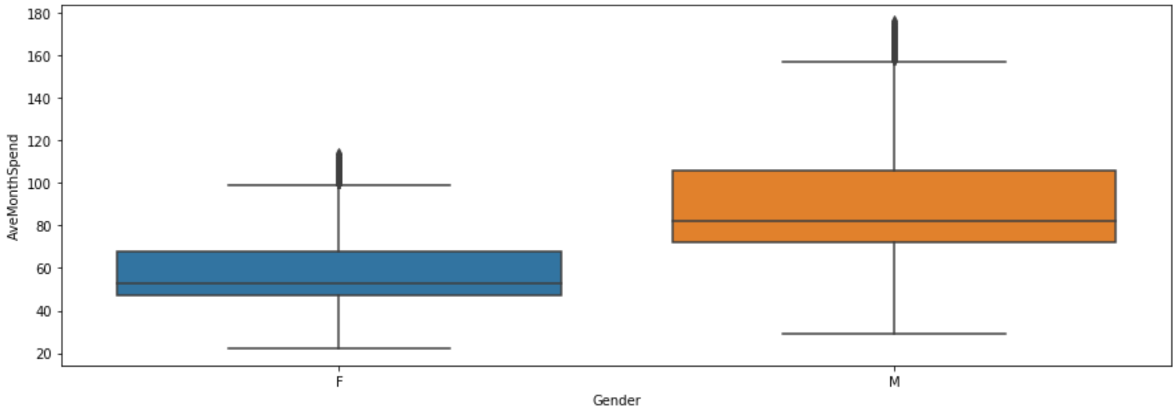
```python
#Checking the effect of each categorical varaible on the target
def plot_box(data, cols, col_y = None):
    for col in cols:
        plt.figure(figsize=(15,5))
        sns.boxplot(y=col_y, x=col, data=data)
        plt.ylabel(col_y) # Set text for the x axis
        plt.xlabel(col)# Set text for y axis
        plt.show()

plot_box(data=train,cols=cat_col,col_y='AveMonthSpend')
```

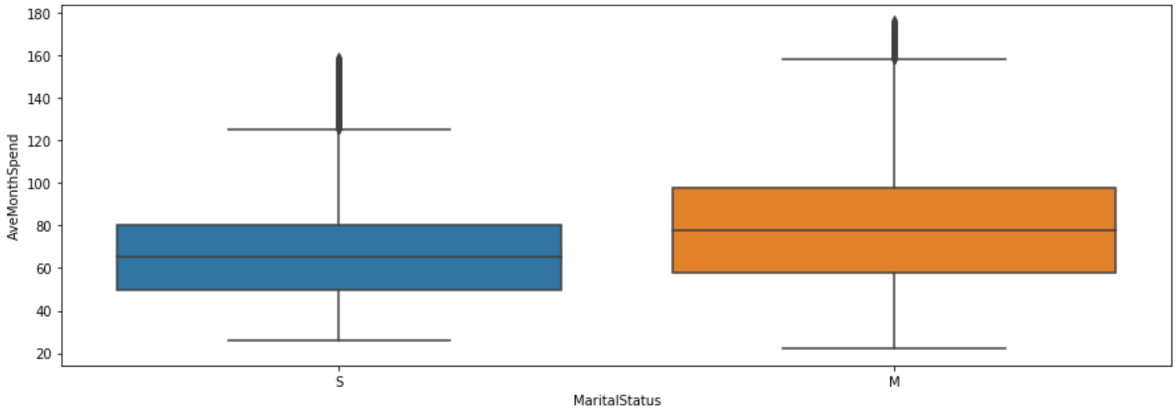
executed in 7.43s, finished 23:58:42 2019-05-08

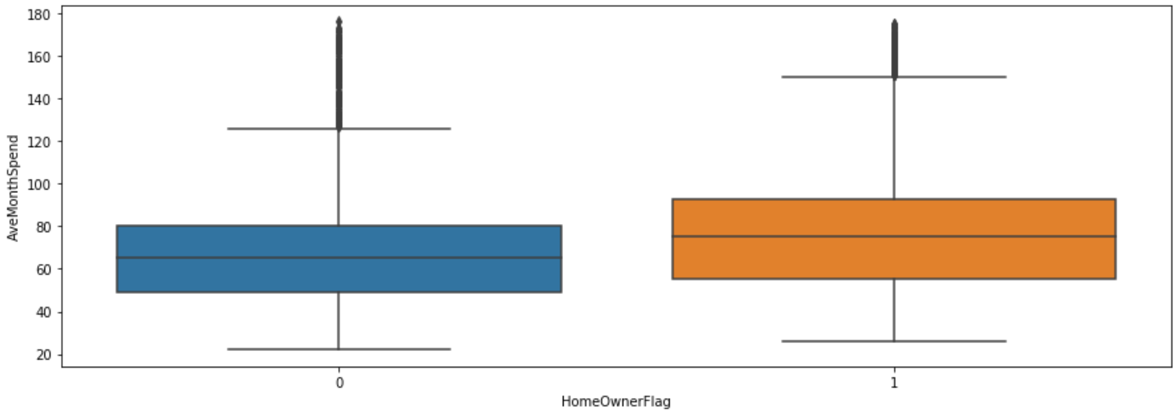


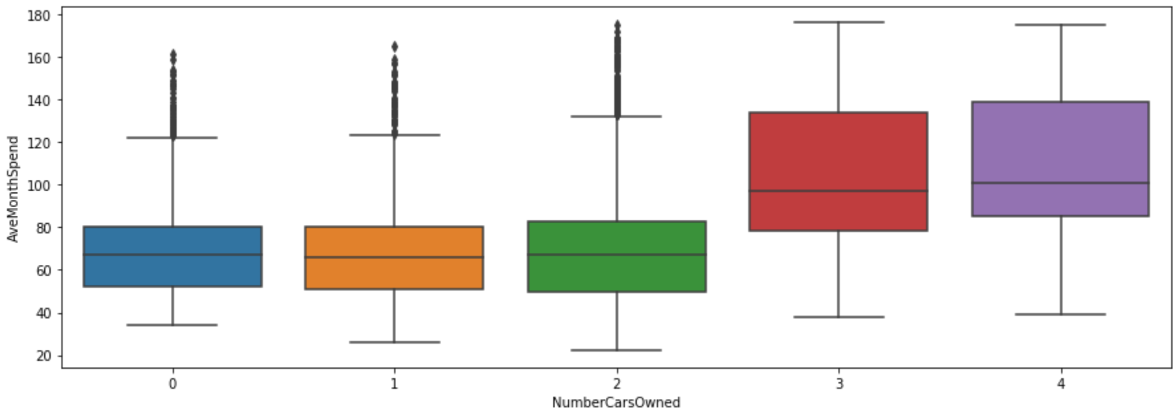


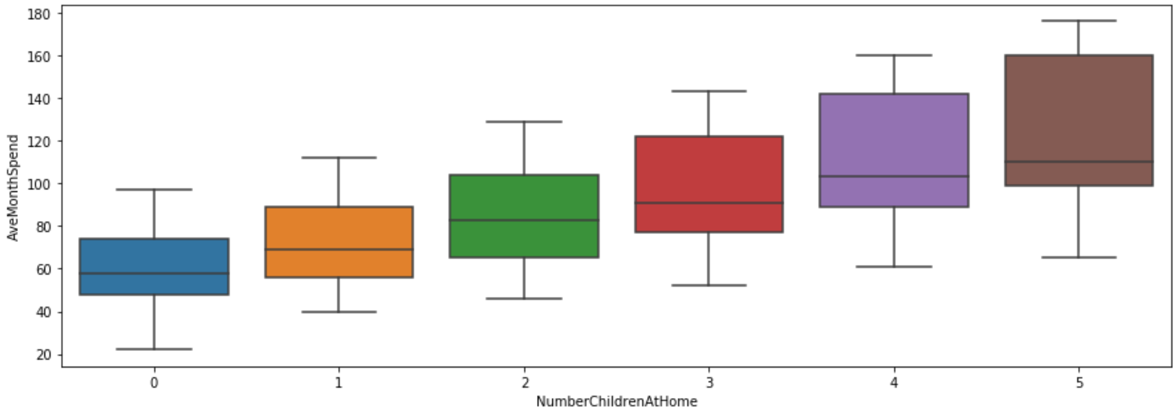


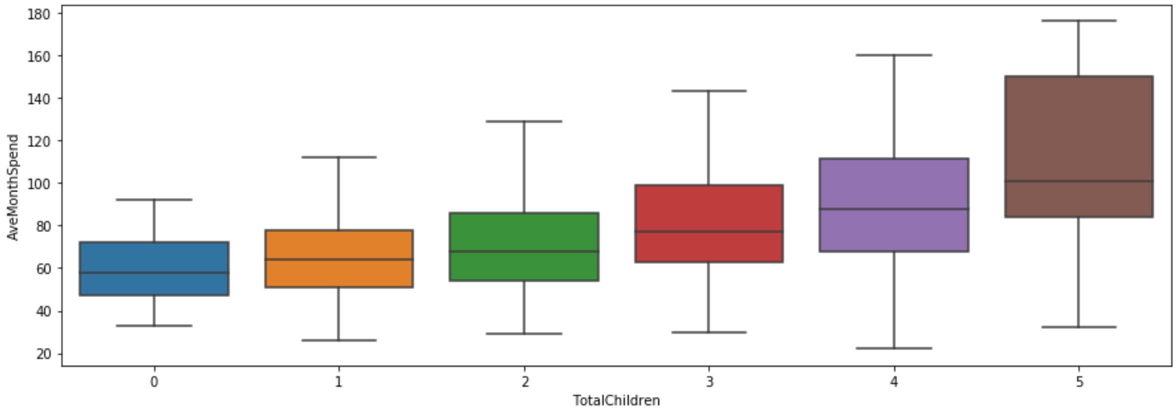

AveMonthSpend
Gender
F
M

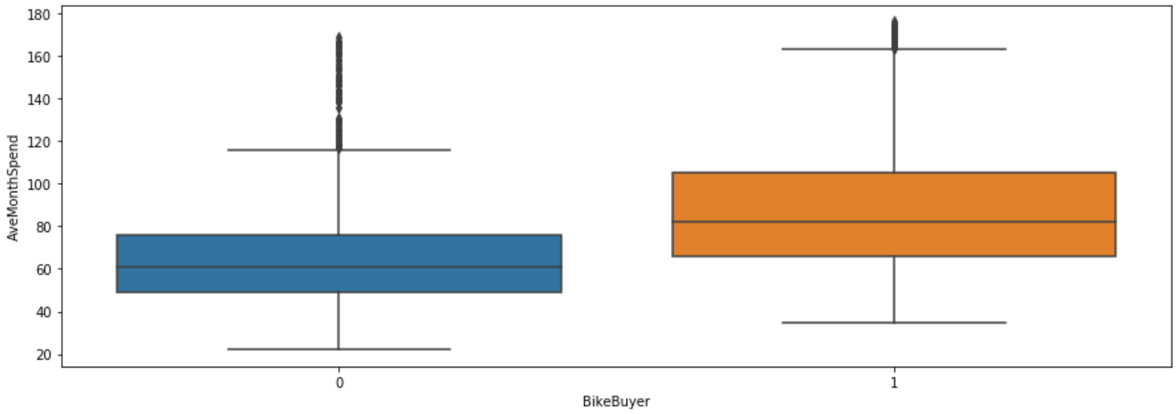

AveMonthSpend
MaritalStatus
S
M


AveMonthSpend
HomeOwnerFlag
0
1


AveMonthSpend
NumberCarsOwned
0
1
2
3
4



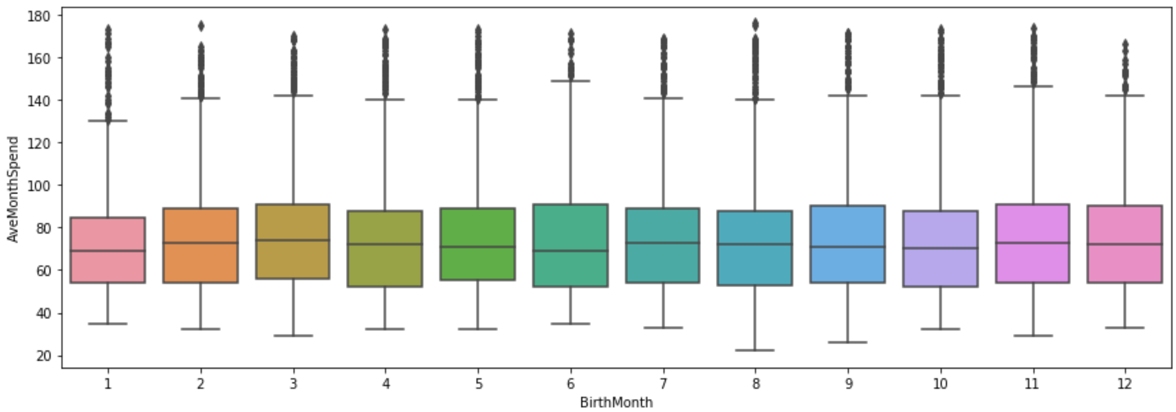




In [27]:

```
#Checking if BirthMonth can be a categorical variable
plot_box(data=train,cols=['BirthMonth'],col_y='AveMonthSpend')
```

executed in 1.11s, finished 23:58:44 2019-05-08



Since birth year has no correlation with average monthly spend, I decided to bin the birth year into ten and test if this would have any correlation as a categorical variable on the target

In [28]:

```
join=train.append(test)
```

executed in 25ms, finished 23:58:44 2019-05-08

In [29]:

```
join['binned_year']=pd.cut(join['BirthYear'],10,labels=[0,1,2,3,4,5,6,7,8,9])
```

executed in 135ms, finished 23:58:44 2019-05-08

In [30]:

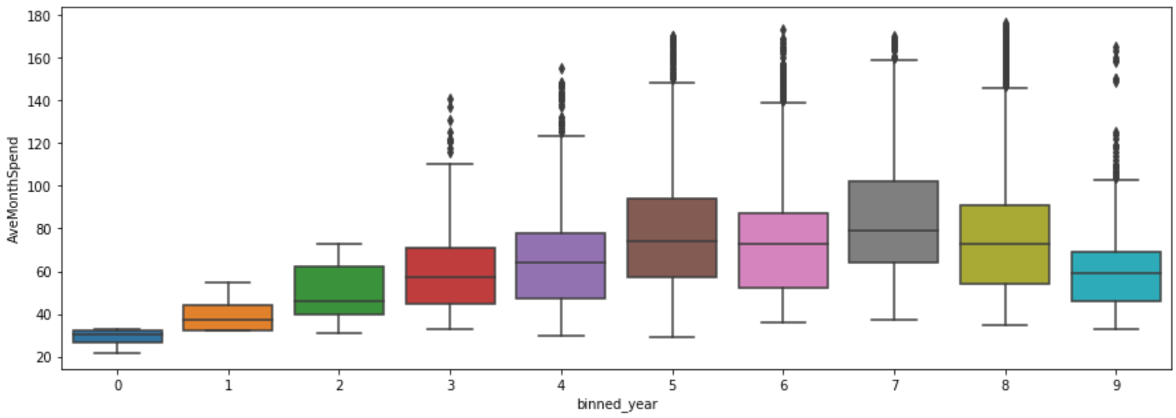
```
train['binned_year']=join['binned_year'][0:7654]
test['binned_year']=join['binned_year'][7654:]
```

executed in 125ms, finished 23:58:44 2019-05-08

In [31]:

```
plot_box(data=train,cols=['binned_year'],col_y='AveMonthSpend')
```

executed in 1.10s, finished 23:58:45 2019-05-08



3 EDA Key Summary

- Looking at the numerical variables, only yearly income correlates with the target
- Looking at the categorical variables, all of them seem to be useful in predicting the target.
- Only Country Region, Education, Occupation, Marital Status and Gender should be encoded. The other categorical variables seem to show some level of ordinality thus they should be used as such.
- Birth month does not seem to be a useful feature for predicting the target variable
- Binned Year seems to be an ordinal categorical variable though the relationship with the target doesn't seem to be entirely linear.

In [ ]:

```

```