In [1]:
```
# Importing the necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import mpl_toolkits
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from sklearn.preprocessing import OneHotEncoder,LabelEncoder
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
import scipy.stats as stats
import warnings
warnings.filterwarnings("ignore")
```

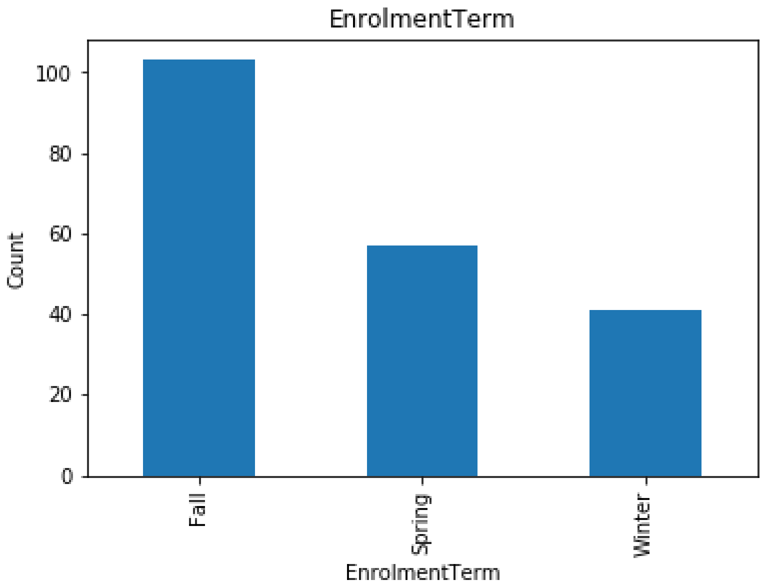
In [2]:
```
#Getting the dataset and checking the variables

data = pd.read_csv('final.csv')
data = data.iloc[:,:19]

# Printing the shape of the dataset

print('Total number of rows and columns are -> ', data.shape)

# Printing the column names of the dataset

print('Headings of the columns of the dataset are -> ',data.columns)
print('Checking the dataset: ')
print(data.head())
```

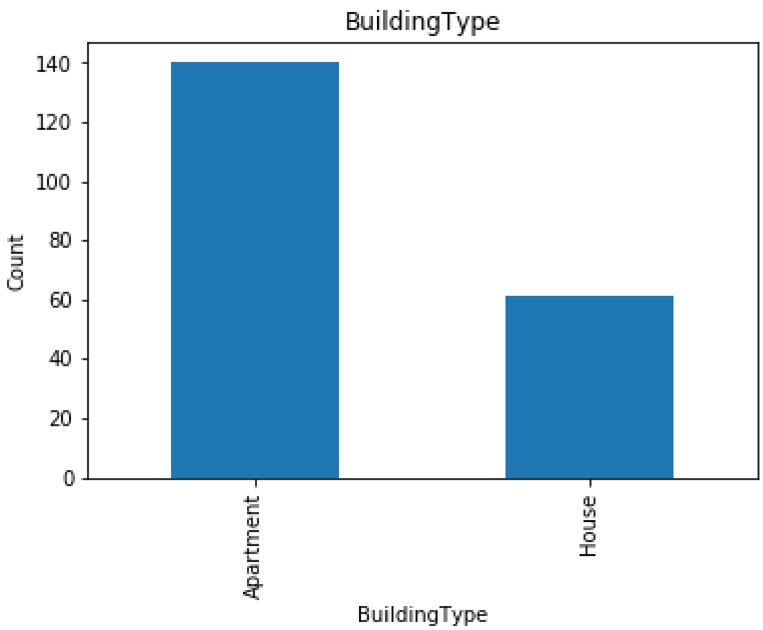
```
Total number of rows and columns are ->  (201, 19)
Headings of the columns of the dataset are ->  Index(['EnrolmentTerm', 'Build
ingType', 'RoomType', 'Furnished',
       'SharedorPrivate', 'HouseAge', 'LocationWard', 'Internet ', 'Hydro',
       'AirConditioning', 'Laundry', 'Parking', 'Terrace ', 'Security',
       'Bedrooms', 'Bathrooms', 'MarketCommute', 'BusStopCommute', 'Rent'],
      dtype='object')
Checking the dataset:
  EnrolmentTerm BuildingType      RoomType Furnished SharedorPrivate HouseAge
\
0          Fall    Apartment  Regular Room       Yes         Private   Middle
1          Fall    Apartment  Regular Room       Yes         Private   Middle
2          Fall        House  Regular Room       Yes          Shared   Middle
3          Fall        House  Regular Room       Yes         Private   Middle
4        Winter    Apartment  Regular Room        No         Private      Old

            LocationWard  Internet    Hydro  AirConditioning  Laundry  Parking
\
0  Central-Columbia ward          1       1                1        1        1
1         Southeast ward          0       1                0        0        0
2  Central-Columbia ward          1       1                1        1        1
3         7- Uptown ward          1       1                1        1        0
4      4- Northeast ward          0       0                0        0        0

   Terrace    Security  Bedrooms  Bathrooms  MarketCommute  BusStopCommute  \
0         0          0         5          2             25               3
1         1          0         7          3             25              18
2         1          0         5          2             15               3
3         0          0         3          1             25               8
4         1          0         1          1             15               3

   Rent
0  2500
1  4900
2  2500
3  2100
4  1300
```

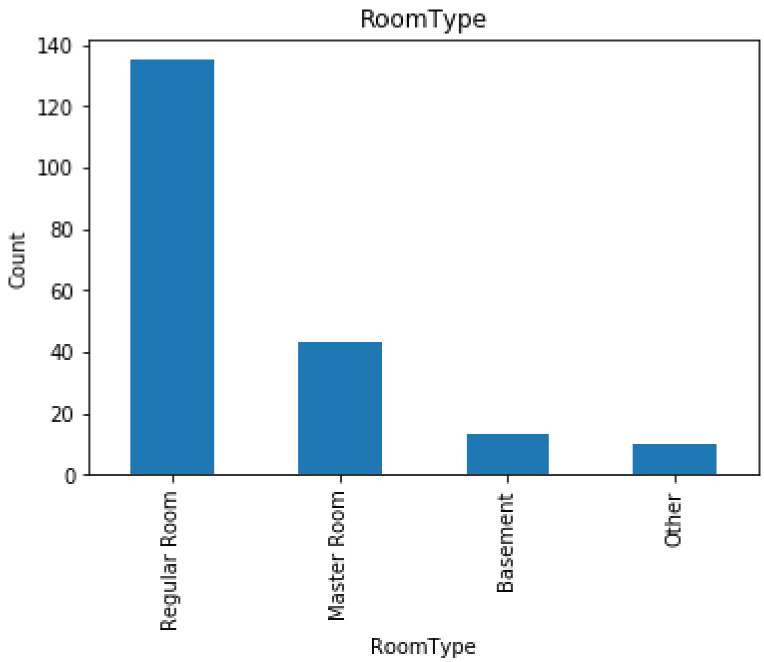
In [4]:
```
# Fetching only catagorical features

catagorical_feature_df = data.select_dtypes(include=['object']).copy()
print(catagorical_feature_df.head())

# Fetching only numerical features

Numerical_feature_df = data.select_dtypes(include=['int64','float64']).copy()
print(Numerical_feature_df.head())
```

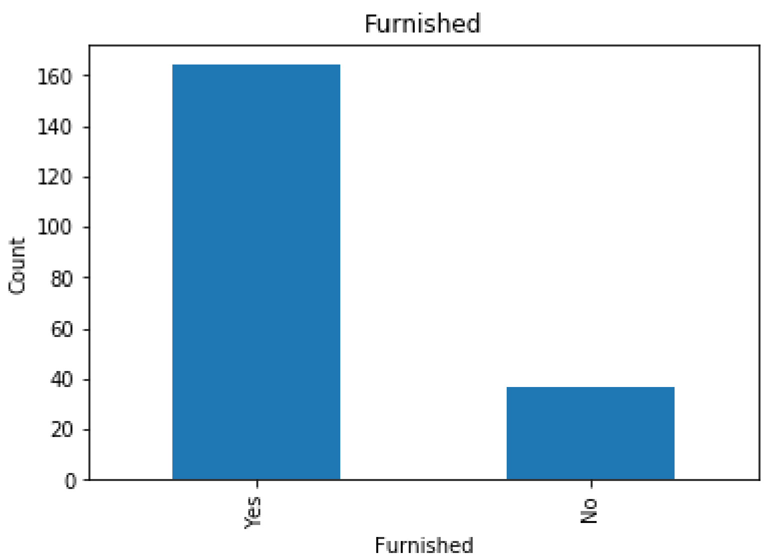
```
  EnrolmentTerm BuildingType      RoomType Furnished SharedorPrivate HouseAge  \
0          Fall    Apartment  Regular Room       Yes         Private   Middle
1          Fall    Apartment  Regular Room       Yes         Private   Middle
2          Fall        House  Regular Room       Yes          Shared   Middle
3          Fall        House  Regular Room       Yes         Private   Middle
4        Winter    Apartment  Regular Room        No         Private      Old

            LocationWard
0  Central-Columbia ward
1         Southeast ward
2  Central-Columbia ward
3         7- Uptown ward
4      4- Northeast ward
   Internet   Hydro  AirConditioning  Laundry  Parking  Terrace   Security  \
0         1       1                1        1        1        0         0
1         0       1                0        0        0        1         0
2         1       1                1        1        1        1         0
3         1       1                1        1        0        0         0
4         0       0                0        0        0        1         0

   Bedrooms  Bathrooms  MarketCommute  BusStopCommute  Rent
0         5          2             25               3  2500
1         7          3             25              18  4900
2         5          2             15               3  2500
3         3          1             25               8  2100
4         1          1             15               3  1300
```

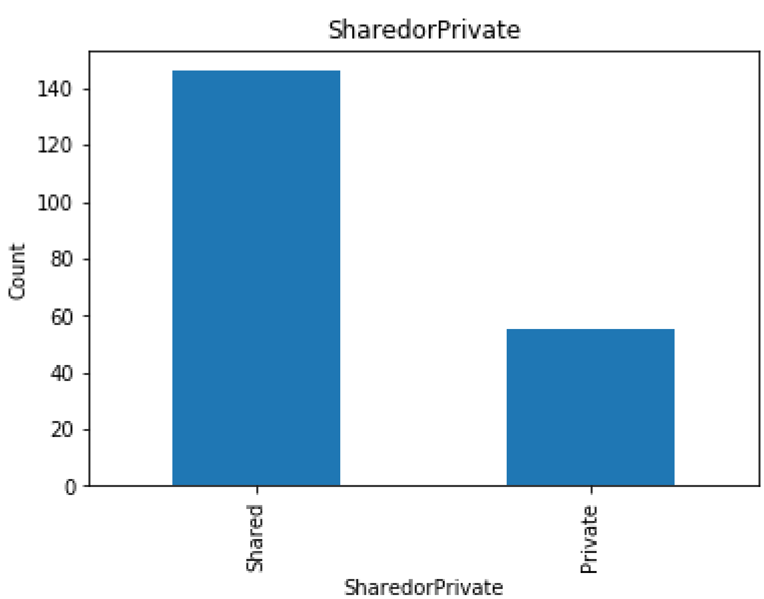
In [5]:
```
# Plots and Graphs

for i in catagorical_feature_df:
    data[i].value_counts().plot(kind='bar')
    plt.title(i)
    plt.xlabel(i)
    plt.ylabel('Count')
    plt.show()
    sns.despine
```

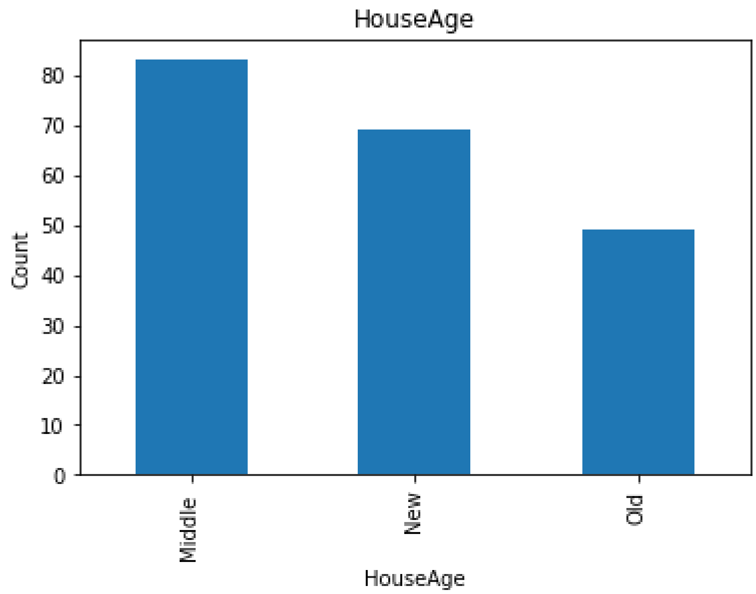

EnrolmentTerm
Count
100
80
60
40
20
0
Fall
Spring
Winter
EnrolmentTerm

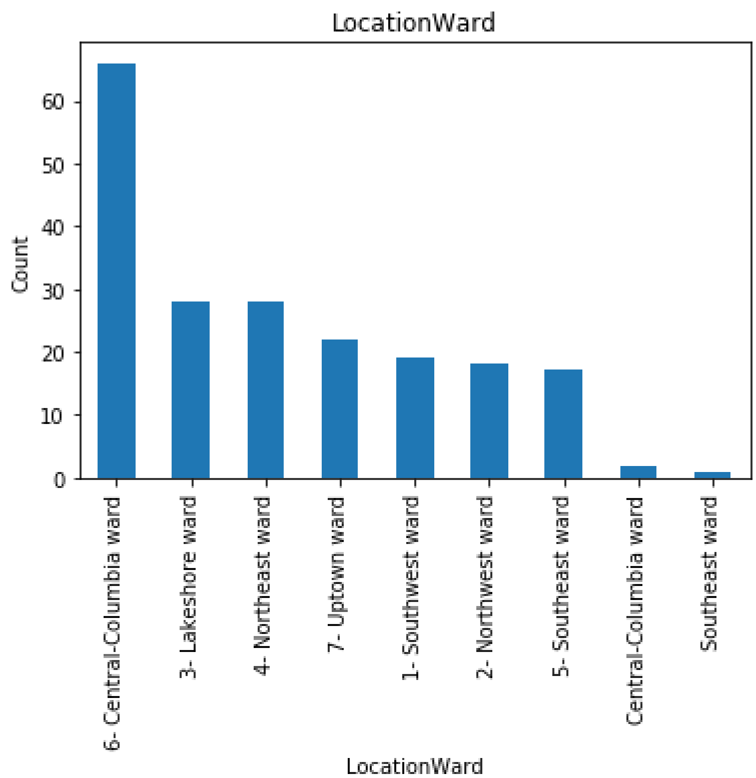

BuildingType
Count
140
120
100
80
60
40
20
0
Apartment
House
BuildingType


RoomType
Count
140
120
100
80
60
40
20
0
Regular Room
Master Room
Basement
Other
RoomType

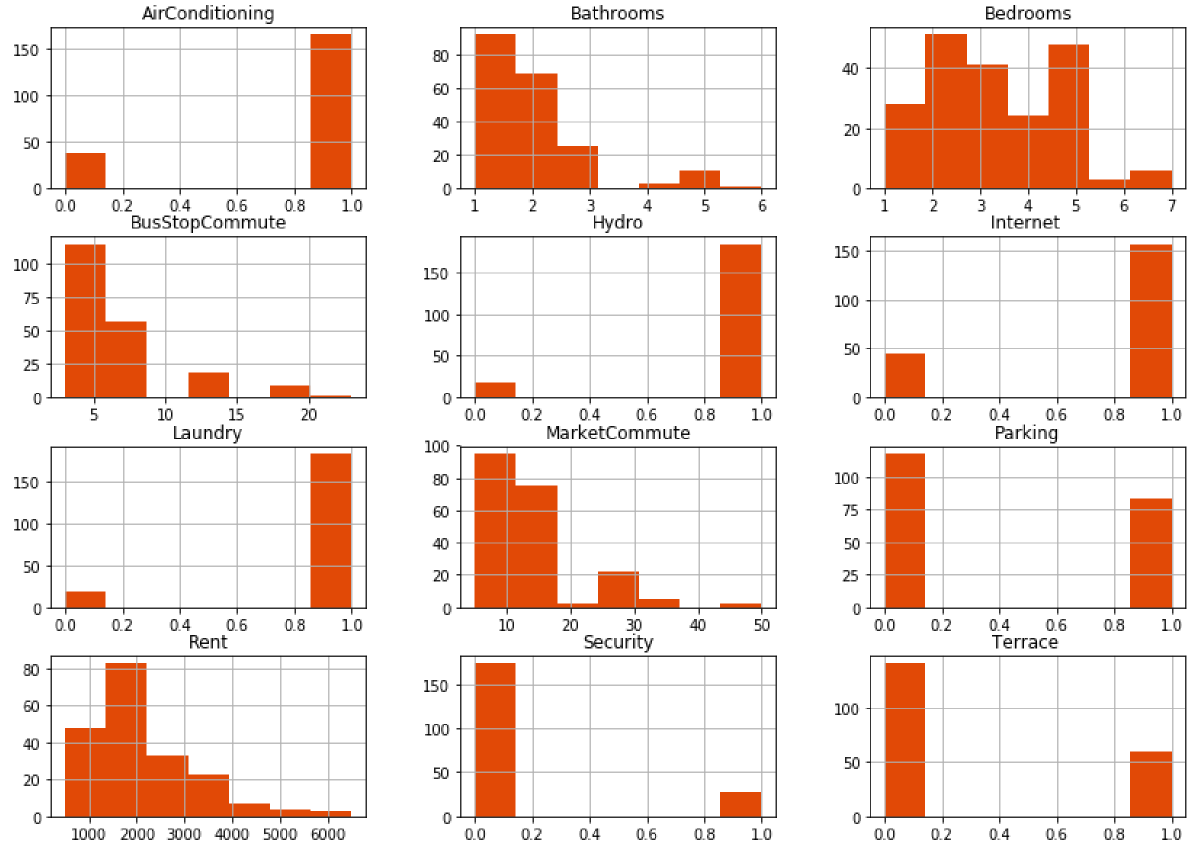

Furnished
Count
160
140
120
100
80
60
40
20
0
Yes
No
Furnished


SharedorPrivate
Count
140
120
100
80
60
40
20
0
Shared
Private
SharedorPrivate

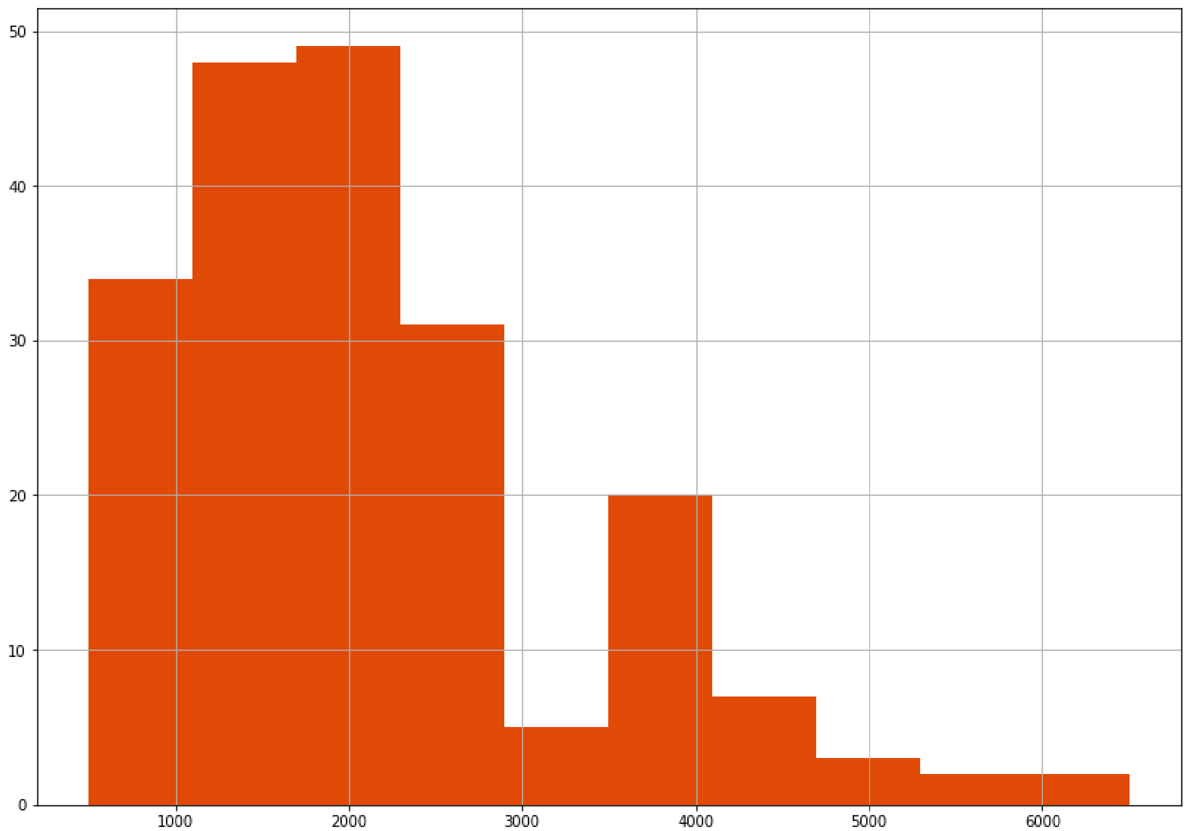

HouseAge
80
70
60
50
40
30
20
10
0
Count
Middle
New
Old
HouseAge

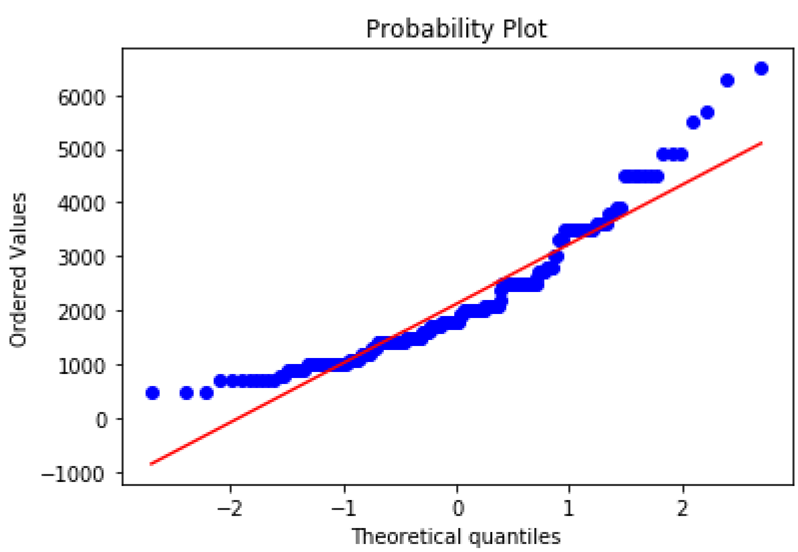

LocationWard
60
50
40
30
20
10
0
Count
6- Central-Columbia ward
3- Lakeshore ward
4- Northeast ward
7- Uptown ward
1- Southwest ward
2- Northwest ward
5- Southeast ward
Central-Columbia ward
Southeast ward
LocationWard

In [6]:
```
# Let's see how the numeric data is distributed.

data.hist(bins=7, figsize=(14,10), color='#E14906')
plt.show()
```

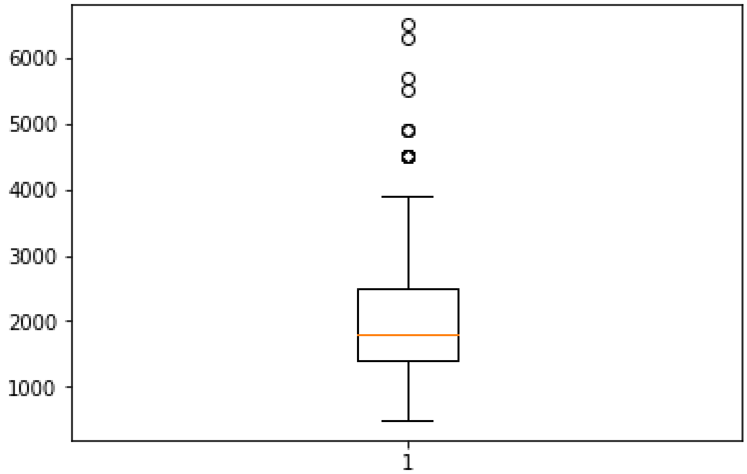


In [7]:
```
data['Rent'].hist(bins=10, figsize=(14,10), color='#E14906')
xlabel = ('Rent')
ylabel = ('Count')
plt.show()
stats.probplot(data['Rent'], dist="norm", plot=plt)
plt.show()
plt.boxplot(data['Rent'])
```



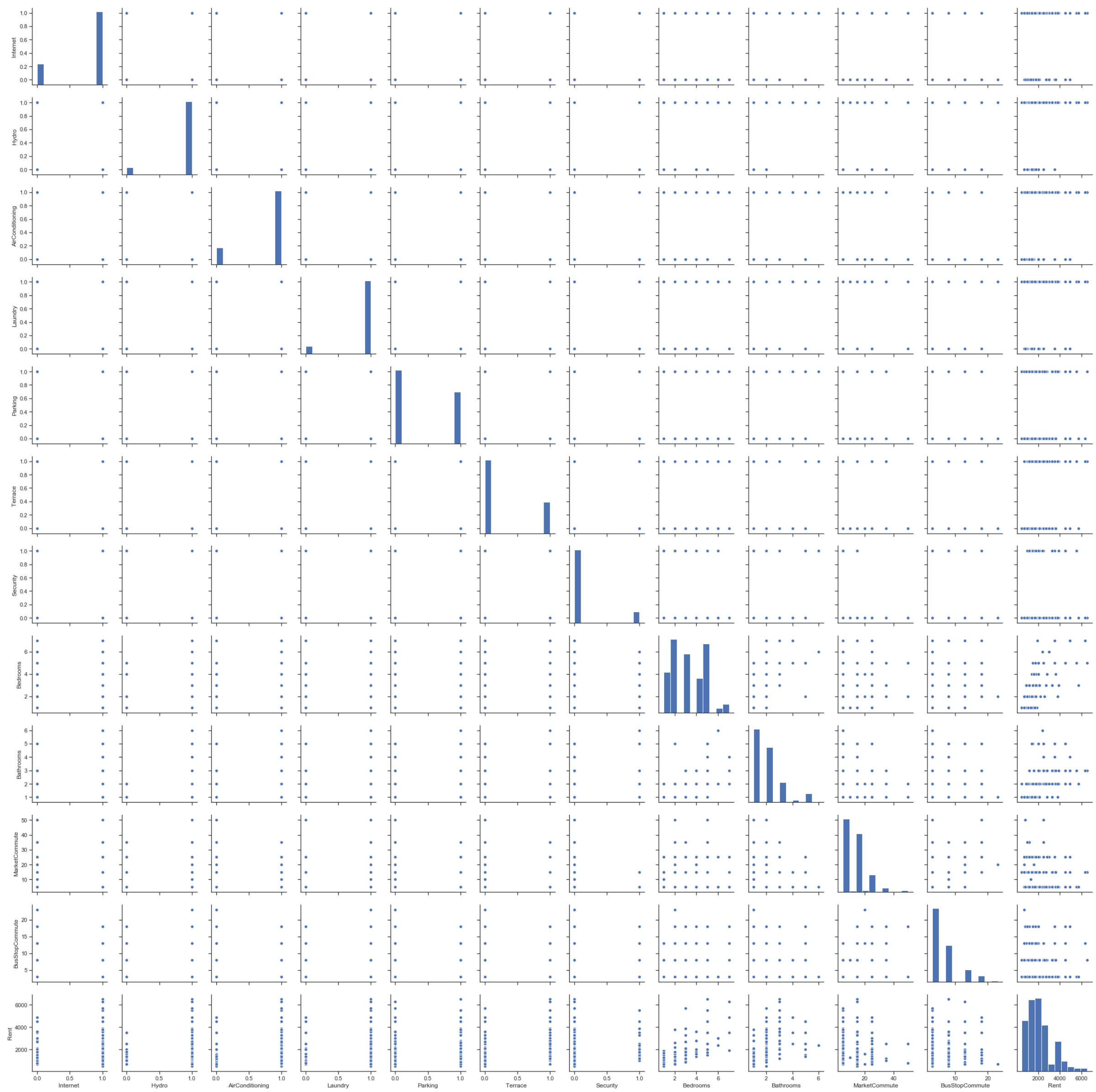


Out[7]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2bae4d02278>,
  <matplotlib.lines.Line2D at 0x2bae4d023c8>],
 'caps': [<matplotlib.lines.Line2D at 0x2bae4d02710>,
  <matplotlib.lines.Line2D at 0x2bae4d02f60>],
 'boxes': [<matplotlib.lines.Line2D at 0x2bae4d02c18>],
 'medians': [<matplotlib.lines.Line2D at 0x2bae4ddbdd8>],
 'fliers': [<matplotlib.lines.Line2D at 0x2bae4ddb7f0>],
 'means': []}
```

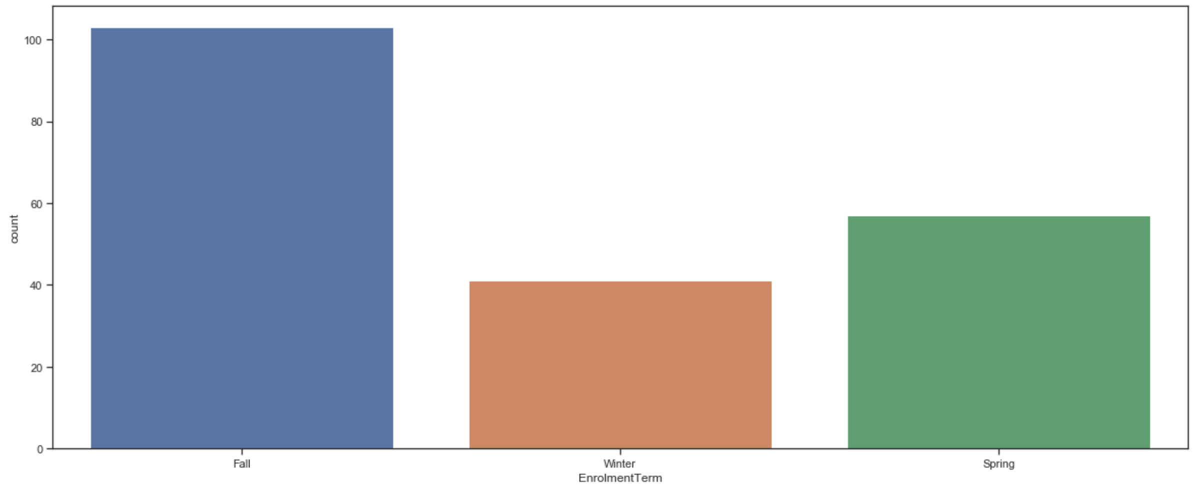

6000
5000
4000
3000
2000
1000
1

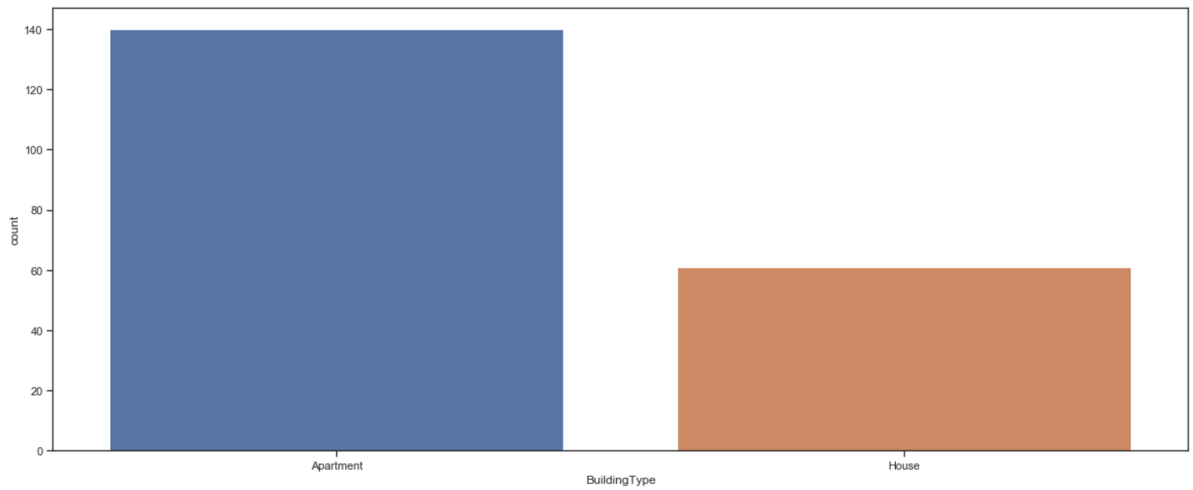
In [8]:
```python
# Initial Plots for dataset
sns.set(style="ticks")
sns.pairplot(data, palette="Set1")
```

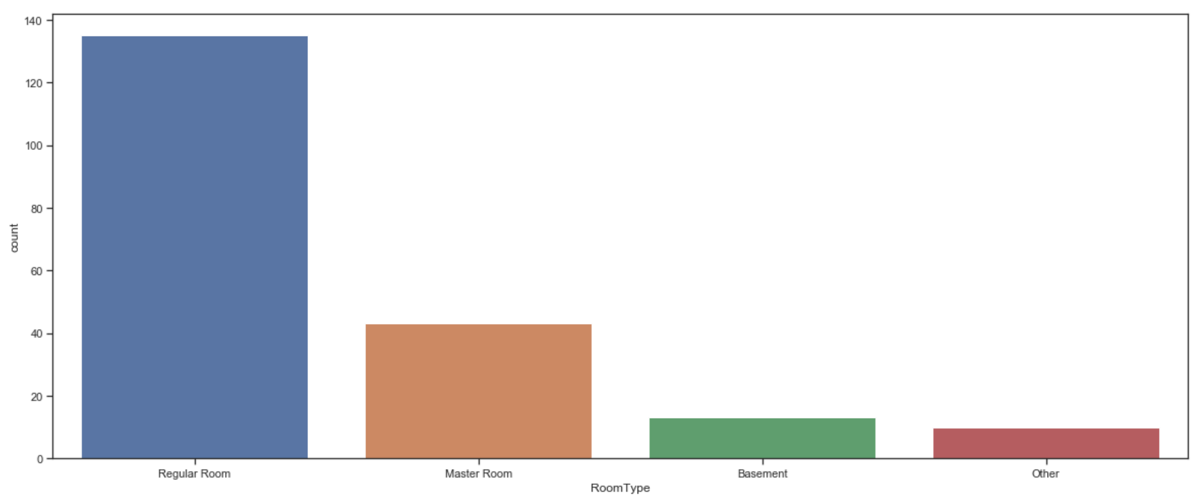
Out[8]: <seaborn.axisgrid.PairGrid at 0x2bae4cea080>

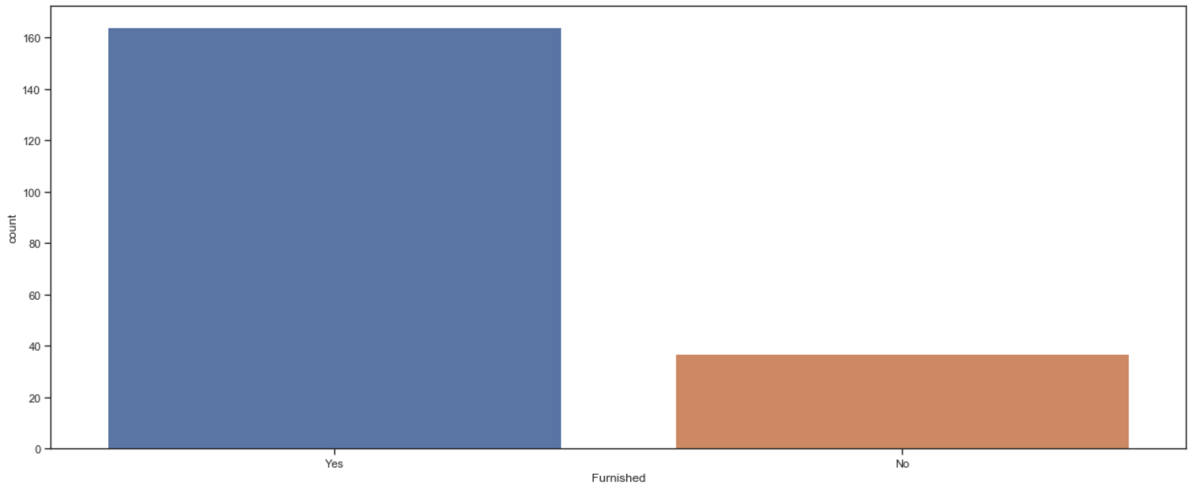


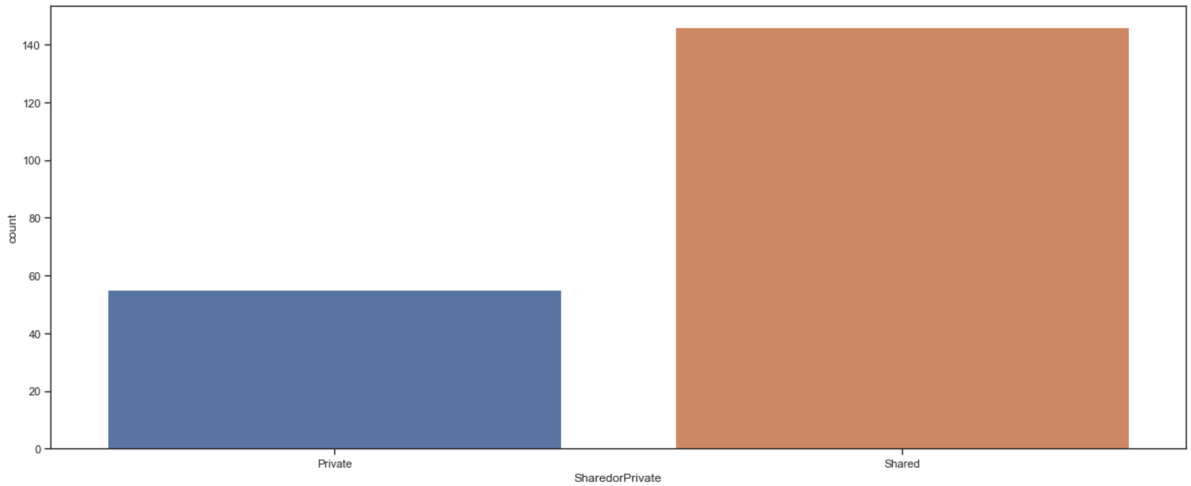
In [9]:
```
# Plotting the catagorical features for dataset
for i in catagorical_feature_df:
    fig = plt.figure(figsize = (20,8))
    sns.countplot(x=i, data=catagorical_feature_df)
    plt.show()
```

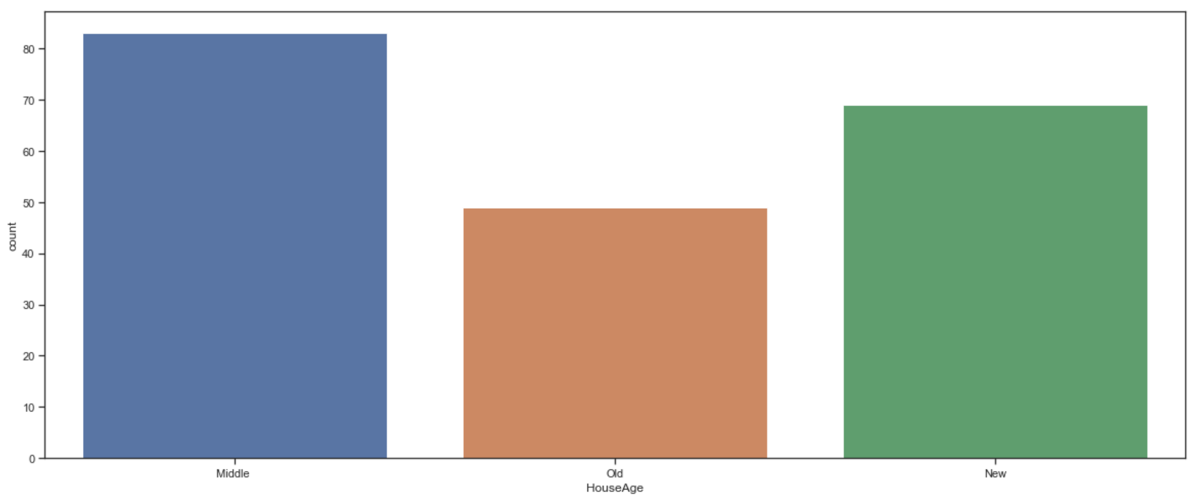

100
80
60
40
20
0
count
Fall
Winter
Spring
EnrolmentTerm

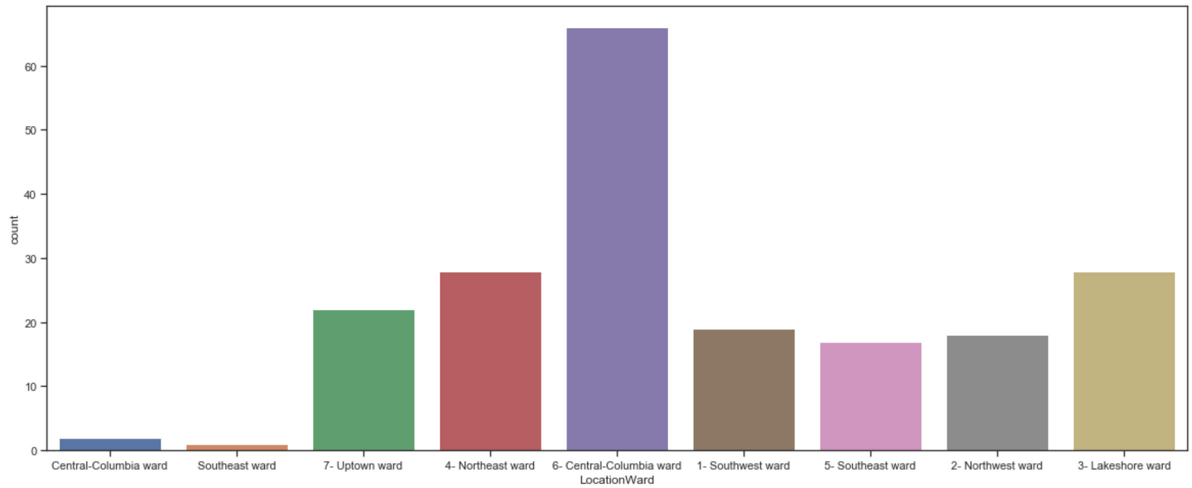

140
120
100
80
60
40
20
0
count
Apartment
House
BuildingType


140
120
100
80
60
40
20
0
count
Regular Room
Master Room
Basement
Other
RoomType


count
160
140
120
100
80
60
40
20
0
Yes
No
Furnished

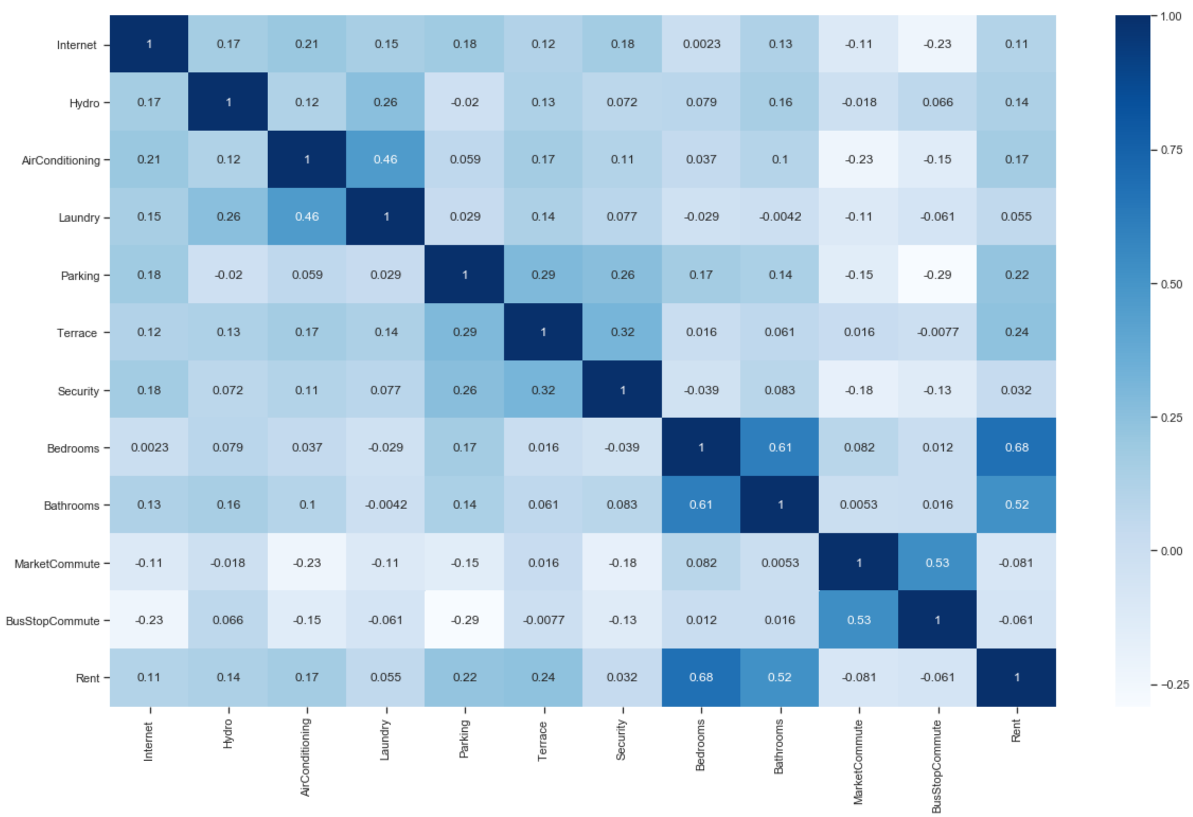

count
140
120
100
80
60
40
20
0
Private
Shared
SharedorPrivate


count
80
70
60
50
40
30
20
10
0
Middle
Old
New
HouseAge



In [10]:
```
# Correlation Plot


#using Pearson correlation (https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b)
plt.figure(figsize=(20,12))

# print(data.head())
cor = data.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Blues)
#plt.show()
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x2baebff23c8>



In [11]:

```
#Correlation with output variable
cor_target = abs(cor["Rent"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.25]
print(relevant_features)
print(cor_target)
```

```
Bedrooms     0.680383
Bathrooms    0.517263
Rent         1.000000
Name: Rent, dtype: float64
Internet           0.110556
Hydro              0.136252
AirConditioning    0.168614
Laundry            0.054797
Parking            0.223150
Terrace            0.241529
Security           0.032367
Bedrooms           0.680383
Bathrooms          0.517263
MarketCommute      0.081427
BusStopCommute     0.060626
Rent               1.000000
Name: Rent, dtype: float64
```

In [12]:
```
#Creating Dummy Variables for categorical features

l = ["EnrolmentTerm","BuildingType","RoomType","Furnished","SharedorPrivate",
"HouseAge","LocationWard"]
one_hot = pd.get_dummies(data[l])
data = data.drop(l,axis = 1)
data = one_hot.join(data)
print(data.head())
print(data.columns)
```

```
   EnrolmentTerm_Fall  EnrolmentTerm_Spring  EnrolmentTerm_Winter  \
0                   1                     0                     0   
1                   1                     0                     0   
2                   1                     0                     0   
3                   1                     0                     0   
4                   0                     0                     1   

   BuildingType_Apartment  BuildingType_House  RoomType_Basement  \
0                       1                   0                  0   
1                       1                   0                  0   
2                       0                   1                  0   
3                       0                   1                  0   
4                       1                   0                  0   

   RoomType_Master Room  RoomType_Other  RoomType_Regular Room  Furnished_No  
\
0                     0               0                      1             0   
1                     0               0                      1             0   
2                     0               0                      1             0   
3                     0               0                      1             0   
4                     0               0                      1             1   

   ...  AirConditioning  Laundry  Parking  Terrace   Security  Bedrooms  \
0  ...                1        1        1         0         0         5   
1  ...                0        0        0         1         0         7   
2  ...                1        1        1         1         0         5   
3  ...                1        1        0         0         0         3   
4  ...                0        0        0         1         0         1   

   Bathrooms  MarketCommute  BusStopCommute  Rent  
0          2             25               3  2500  
1          3             25              18  4900  
2          2             15               3  2500  
3          1             25               8  2100  
4          1             15               3  1300  

[5 rows x 37 columns]
Index(['EnrolmentTerm_Fall', 'EnrolmentTerm_Spring', 'EnrolmentTerm_Winter',
       'BuildingType_Apartment', 'BuildingType_House', 'RoomType_Basement',
       'RoomType_Master Room', 'RoomType_Other', 'RoomType_Regular Room',
       'Furnished_No', 'Furnished_Yes', 'SharedorPrivate_Private',
       'SharedorPrivate_Shared', 'HouseAge_Middle ', 'HouseAge_New',
       'HouseAge_Old', 'LocationWard_1- Southwest ward',
       'LocationWard_2- Northwest ward', 'LocationWard_3- Lakeshore ward',
       'LocationWard_4- Northeast ward', 'LocationWard_5- Southeast ward',
       'LocationWard_6- Central-Columbia ward', 'LocationWard_7- Uptown war
d',
       'LocationWard_Central-Columbia ward', 'LocationWard_Southeast ward',
       'Internet ', 'Hydro', 'AirConditioning', 'Laundry', 'Parking',
       'Terrace ', 'Security', 'Bedrooms', 'Bathrooms', 'MarketCommute',
       'BusStopCommute', 'Rent'],
      dtype='object')
```

In [13]:
```
data.head()

# Everything looks proper now, encoded and all dummy variables have been creat
ed
```

Out[13]:

	EnrolmentTerm_Fall	EnrolmentTerm_Spring	EnrolmentTerm_Winter	BuildingType_Apartment	B
0	1	0	0	1	
1	1	0	0	1	
2	1	0	0	0	
3	1	0	0	0	
4	0	0	1	1	

5 rows × 37 columns

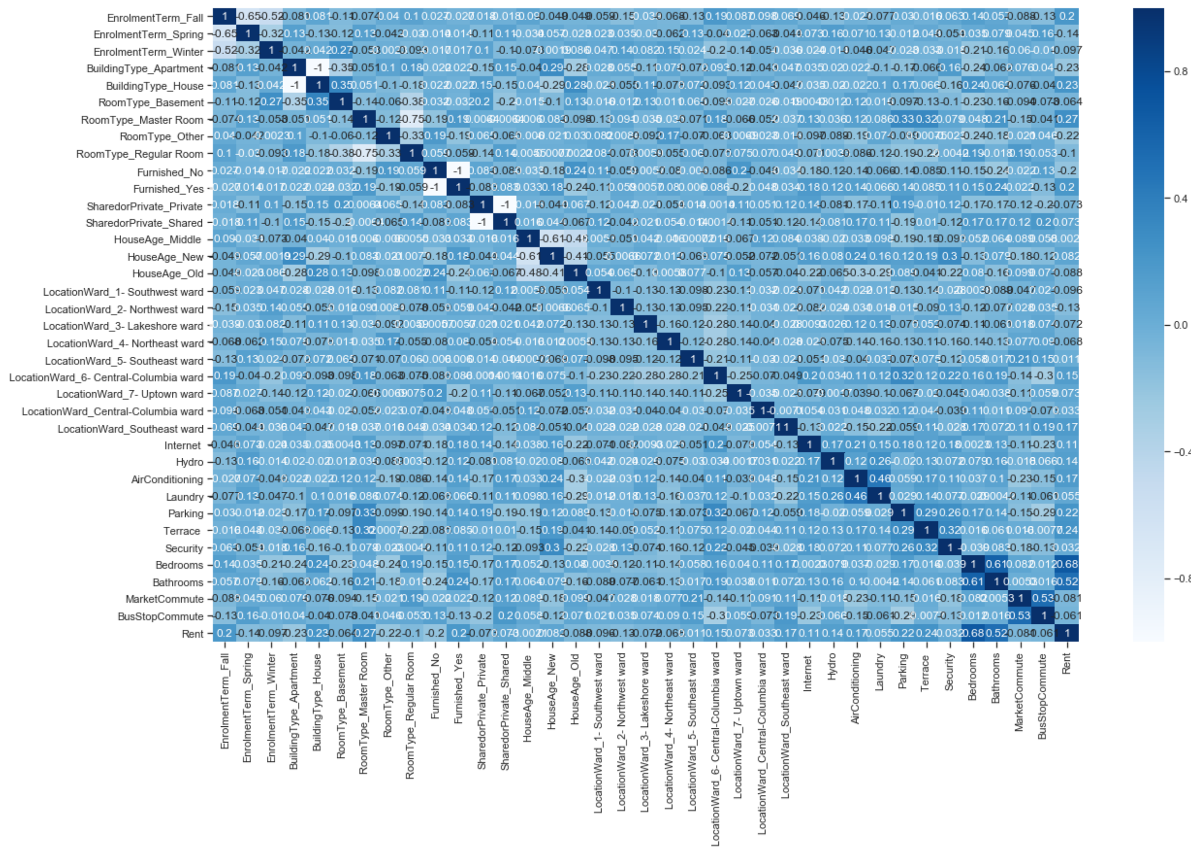
In [ ]:
```
'''
Variables within a dataset can be related for lots of reasons.

For example:
One variable could cause or depend on the values of another variable.
One variable could be lightly associated with another variable.
Two variables could depend on a third unknown variable.

The Pearson correlation coefficient (named for Karl Pearson) can be used to su
mmarize the strength of the linear relationship between two data samples.

The Pearson’s correlation coefficient is calculated as the covariance of the t
wo variables divided by the product of the standard deviation of each data sam
ple.
It is the normalization of the covariance between the two variables to give an
interpretable score.
'''
```

In [14]:
```
from scipy.stats import pearsonr
corrData = []
for i in data.columns:
    for j in data.columns:
        if i =="Rent" or j == 'Rent':
            break
        corr, _ = pearsonr(data[i],data[j])
        corrData.append(corr)
        if corr >0.5 and corr<1:
            print(i,j,corr)
```

```
Bedrooms Bathrooms 0.6143499893719668
Bathrooms Bedrooms 0.6143499893719668
MarketCommute BusStopCommute 0.5322804637501042
BusStopCommute MarketCommute 0.5322804637501042
```

In [15]:
```python
#using Pearson correlation (https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b)
plt.figure(figsize=(20,12))
# print(data.head())
cor = data.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Blues)
#plt.show()
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x2baecb8cbe0>



In [16]:
```
#Correlation with output variable
cor_target = abs(cor["Rent"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.25]
print(relevant_features)
print(cor_target)
```

```
RoomType_Master Room    0.272630
Bedrooms                0.680383
Bathrooms               0.517263
Rent                    1.000000
Name: Rent, dtype: float64
EnrolmentTerm_Fall                       0.203562
EnrolmentTerm_Spring                     0.139418
EnrolmentTerm_Winter                     0.096558
BuildingType_Apartment                   0.225291
BuildingType_House                       0.225291
RoomType_Basement                        0.064181
RoomType_Master Room                     0.272630
RoomType_Other                           0.216654
RoomType_Regular Room                    0.104142
Furnished_No                             0.198271
Furnished_Yes                            0.198271
SharedorPrivate_Private                  0.073320
SharedorPrivate_Shared                   0.073320
HouseAge_Middle                          0.002140
HouseAge_New                             0.082192
HouseAge_Old                             0.088437
LocationWard_1- Southwest ward           0.095936
LocationWard_2- Northwest ward           0.129417
LocationWard_3- Lakeshore ward           0.071789
LocationWard_4- Northeast ward           0.068032
LocationWard_5- Southeast ward           0.010615
LocationWard_6- Central-Columbia ward    0.154015
LocationWard_7- Uptown ward              0.073286
LocationWard_Central-Columbia ward       0.032676
LocationWard_Southeast ward              0.170993
Internet                                 0.110556
Hydro                                    0.136252
AirConditioning                          0.168614
Laundry                                  0.054797
Parking                                  0.223150
Terrace                                  0.241529
Security                                 0.032367
Bedrooms                                 0.680383
Bathrooms                                0.517263
MarketCommute                            0.081427
BusStopCommute                           0.060626
Rent                                     1.000000
Name: Rent, dtype: float64
```

In [49]:
```
#Setting up the dependent and independent variables

data1 = data[['Bedrooms','RoomType_Master Room', 'Rent']]
X = data1.drop('Rent', axis=1)

y = data.iloc[:, 2:3].values
```

In [ ]:

In [18]:
```
# Let's begin prediction
```

In [19]:
```
from sklearn.model_selection import train_test_split
X_Train, X_Test, y_Train, y_Test = train_test_split(X, y, test_size = 0.2, ran
dom_state = 0)
from sklearn.metrics import mean_squared_error
```

In [20]:
```
# 1 - Multiple Linear regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_Train, y_Train)

# Predicting the test result
y_pred = regressor.predict(X_Test)
print(y_pred)
regressor.score(X_Test,y_Test)
```

```
[2807.4369722  1867.72218002 2337.57957611 3483.87438156  928.00738784
 3747.15176438 2807.4369722  1397.86478393 2807.4369722   928.00738784
 1867.72218002 3747.15176438 1397.86478393 2074.3021933  3483.87438156
 1397.86478393 1397.86478393 3014.01698548 1397.86478393 3483.87438156
  928.00738784 1604.44479721 2337.57957611 1867.72218002 3747.15176438
 1867.72218002  928.00738784 1397.86478393 1867.72218002 2807.4369722
 2337.57957611 2807.4369722  1867.72218002 2544.15958939 3483.87438156
 1867.72218002 2544.15958939  928.00738784 2807.4369722  1397.86478393
 1397.86478393]
```

Out[20]:
```
0.6218912987272593
```

In [21]:
```
# Defining a general funciton

def get_score(model, X_Train, X_Test, y_Train, y_Test):
    model.fit(X_Train, y_Train)
    return model.score(X_Test, y_Test)

get_score(regressor,X_Train, X_Test, y_Train, y_Test)
```

Out[21]:
```
0.6218912987272593
```

In [23]:
```
# Improvising the training and testing dataset by applying KFold cross validat
ion

from sklearn.model_selection import KFold
folds = KFold(n_splits=6)
scores_linear = []

for train_index, test_index in folds.split(X,y):
    X_Train, X_Test = X.iloc[train_index], X.iloc[test_index]
    y_Train, y_Test = y[train_index], y[test_index]
    scores_linear.append(get_score(regressor, X_Train, X_Test, y_Train, y_Test
))

scores_linear
np.max(scores_linear)
```

Out[23]:
```
0.6469315919144781
```

In [ ]:
```
# So all the training and testing data is now selected using KFold cross Valid
ation.
```

In [25]:
```
from sklearn.model_selection import cross_val_score
print(max(cross_val_score(regressor, X, y,cv=10)))
```

```
0.6842290486631266
```

In [26]:
```
# 2 - Decision Tree Regression

from sklearn.tree import DecisionTreeRegressor
Dregressor = DecisionTreeRegressor(random_state = 0)
Dregressor.fit(X,y)
```

Out[26]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=0, splitter='best')
```

In [51]:
```
y_pred = Dregressor.predict(X_Test)
df=pd.DataFrame({'Actual':y_Test, 'Predicted':y_pred})
df
```

Out[51]:

	Actual	Predicted
0	1400	1328.260870
1	1800	1328.260870
2	1500	1841.129032
3	1800	1841.129032
4	1800	1328.260870
5	1200	1841.129032
6	2100	1841.129032
7	2700	1841.129032
8	1400	1328.260870
9	1100	1109.523810
10	1500	1841.129032
11	700	1109.523810
12	1400	1328.260870
13	1200	1841.129032
14	6300	4404.166667
15	1700	1109.523810
16	3300	2700.000000
17	1800	1328.260870
18	2500	3678.571429
19	4500	3678.571429
20	1800	1328.260870
21	1500	1841.129032
22	2100	1841.129032
23	1800	1328.260870
24	800	1328.260870
25	1200	1841.129032
26	1800	1328.260870
27	1300	1109.523810
28	700	1328.260870
29	1400	1328.260870
30	1700	1109.523810
31	1400	1328.260870
32	1000	1328.260870

In [30]:
```
# Checking the accuracy
```

In [84]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_Test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_Test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_Test, y
_pred)))
```

```
Mean Absolute Error: 0.35392129487382884
Mean Squared Error: 0.22063429454028347
Root Mean Squared Error: 0.4697172495664636
```

In [86]:
```
from sklearn.model_selection import train_test_split
X_Train, X_Test, y_Train, y_Test = train_test_split(X, y, test_size = 0.2, ran
dom_state = 0)
from sklearn.metrics import mean_squared_error
```

In [87]:
```
# 3 - Random Forest Regression

from sklearn.ensemble import RandomForestRegressor
RFregressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
RFregressor.fit(X,y)
```

Out[87]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [93]:
```
y_pred1 = RFregressor.predict(X_Test)
```

In [96]:
```
# Checking the accuracy
```

In [94]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_Test, y_pred1))
print('Mean Squared Error:', metrics.mean_squared_error(y_Test, y_pred1))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_Test, y
_pred1)))
```

```
Mean Absolute Error: 0.3381566049744372
Mean Squared Error: 0.19548020586372303
Root Mean Squared Error: 0.4421314350549201
```

In [95]:
```
# This shows that Random forest regressor gives comparatively less mean square
error than Decision tree regressor.
```

In [ ]:
```
# End of this file
```