In [28]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
import matplotlib.pyplot as plt
import seaborn as sns
```

In [29]:

```
data = pd.read_csv("C:/Users/Atharva/OneDrive/Desktop/House_Rent_Dataset2.csv")
print(data.head())
```

```
   Posted On  BHK   Rent  Size            Floor    Area Type  \
0  2022-05-18    2  10000  1100  Ground out of 2   Super Area   
1  2022-05-13    2  20000   800       1 out of 3   Super Area   
2  2022-05-16    2  17000  1000       1 out of 3   Super Area   
3  2022-07-04    2  10000   800       1 out of 2   Super Area   
4  2022-05-09    2   7500   850       1 out of 2  Carpet Area   

             Area Locality     City Furnishing Status  Tenant Preferred  \
0                   Bandel  Kolkata       Unfurnished  Bachelors/Family   
1  Phool Bagan, Kankurgachi  Kolkata    Semi-Furnished  Bachelors/Family   
2   Salt Lake City Sector 2  Kolkata    Semi-Furnished  Bachelors/Family   
3              Dumdum Park  Kolkata       Unfurnished  Bachelors/Family   
4            South Dum Dum  Kolkata       Unfurnished         Bachelors   

   Bathroom Point of Contact  
0         2    Contact Owner  
1         1    Contact Owner  
2         1    Contact Owner  
3         1    Contact Owner  
4         1    Contact Owner  
```

In [30]:

```
# Pre-process dataset
data['Posted On'] = pd.to_datetime(data['Posted On'])
data['Posted Month'] = data['Posted On'].dt.month
```

In [31]:

```
# Handle 'Floor' column with additional non-numeric cases
def extract_floor_level(floor):
    try:
        if 'Ground' in floor:
            return 0
        elif 'Upper' in floor:
            return -1  # Using -1 for 'Upper' floors or any unknown case
        return int(floor.split(' ')[0])
    except ValueError:
        return -1  # For any other unexpected non-numeric entries

data['Floor Level'] = data['Floor'].apply(extract_floor_level)
```

In [32]:

```
# Encoding categorical variables
data['Area Type'] = data['Area Type'].map({'Super Area': 1, 'Carpet Area': 2, 'Built Area': 3})
data['Furnishing Status'] = data['Furnishing Status'].map({'Unfurnished': 0, 'Semi-Furnished': 1, 'Furnished': 2
data['Tenant Preferred'] = data['Tenant Preferred'].map({'Bachelors': 1, 'Bachelors/Family': 2, 'Family': 3})
data['City'] = data['City'].factorize()[0]
```

In [36]:

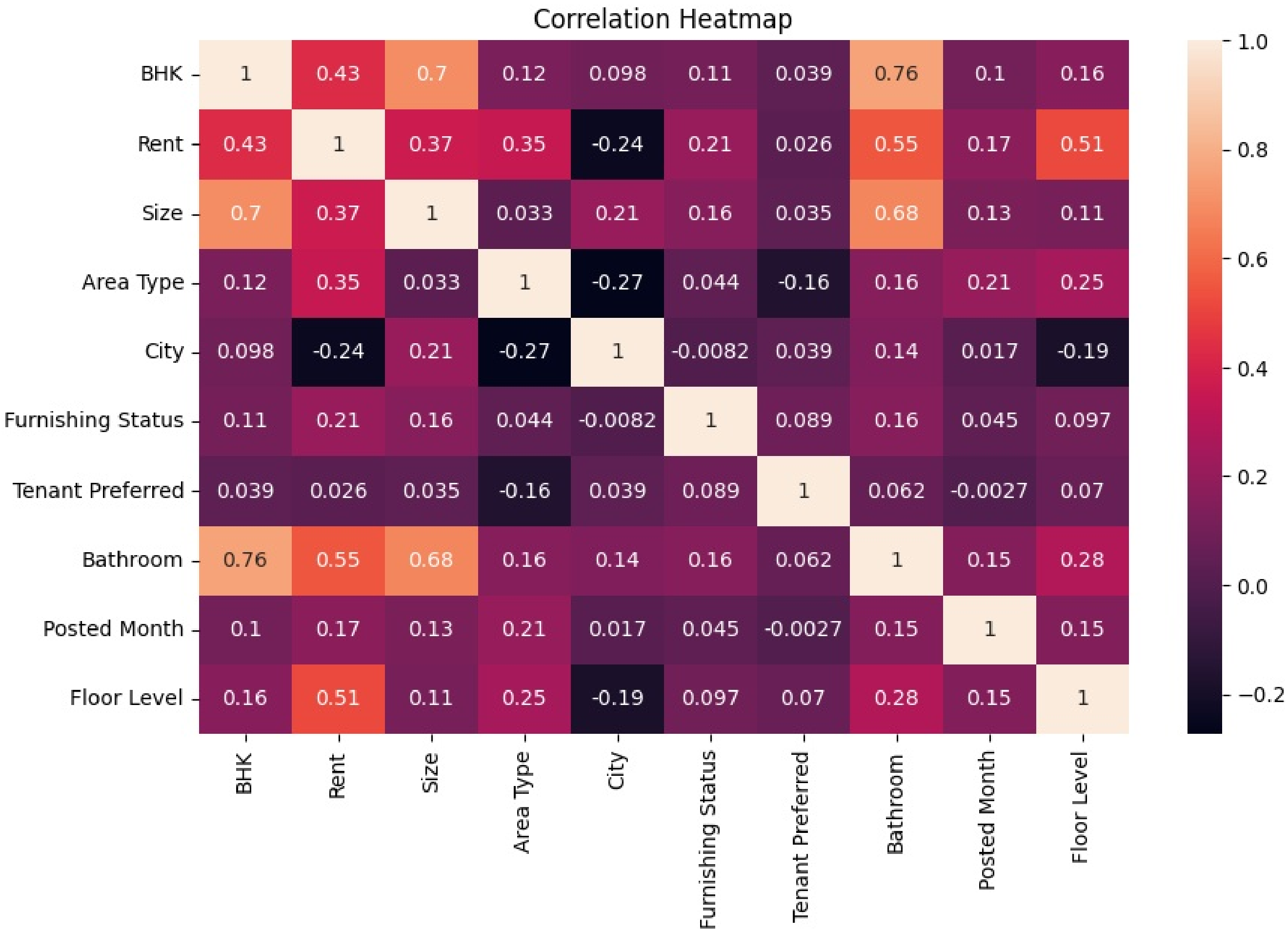
```
# Drop unused columns
data.drop(columns=['Posted On', 'Floor', 'Area Locality', 'Point of Contact'], inplace=True)
```

In [37]:

```
# Identify and remove outliers based on Rent and Size
data = data[(data['Rent'] < data['Rent'].quantile(0.99)) & (data['Size'] < data['Size'].quantile(0.99))]
```

In [38]:

```
# Check correlation
plt.figure(figsize=(10, 6))
sns.heatmap(data.corr(), annot=True)
plt.title("Correlation Heatmap")
plt.show()
```



	BHK	Rent	Size	Area Type	City	Furnishing Status	Tenant Preferred	Bathroom	Posted Month	Floor Level
BHK	1	0.43	0.7	0.12	0.098	0.11	0.039	0.76	0.1	0.16
Rent	0.43	1	0.37	0.35	-0.24	0.21	0.026	0.55	0.17	0.51
Size	0.7	0.37	1	0.033	0.21	0.16	0.035	0.68	0.13	0.11
Area Type	0.12	0.35	0.033	1	-0.27	0.044	-0.16	0.16	0.21	0.25
City	0.098	-0.24	0.21	-0.27	1	-0.0082	0.039	0.14	0.017	-0.19
Furnishing Status	0.11	0.21	0.16	0.044	-0.0082	1	0.089	0.16	0.045	0.097
Tenant Preferred	0.039	0.026	0.035	-0.16	0.039	0.089	1	0.062	-0.0027	0.07
Bathroom	0.76	0.55	0.68	0.16	0.14	0.16	0.062	1	0.15	0.28
Posted Month	0.1	0.17	0.13	0.21	0.017	0.045	-0.0027	0.15	1	0.15
Floor Level	0.16	0.51	0.11	0.25	-0.19	0.097	0.07	0.28	0.15	1

In [39]:
```
# Prepare data for training
X = data.drop(columns=['Rent'])
y = data['Rent']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

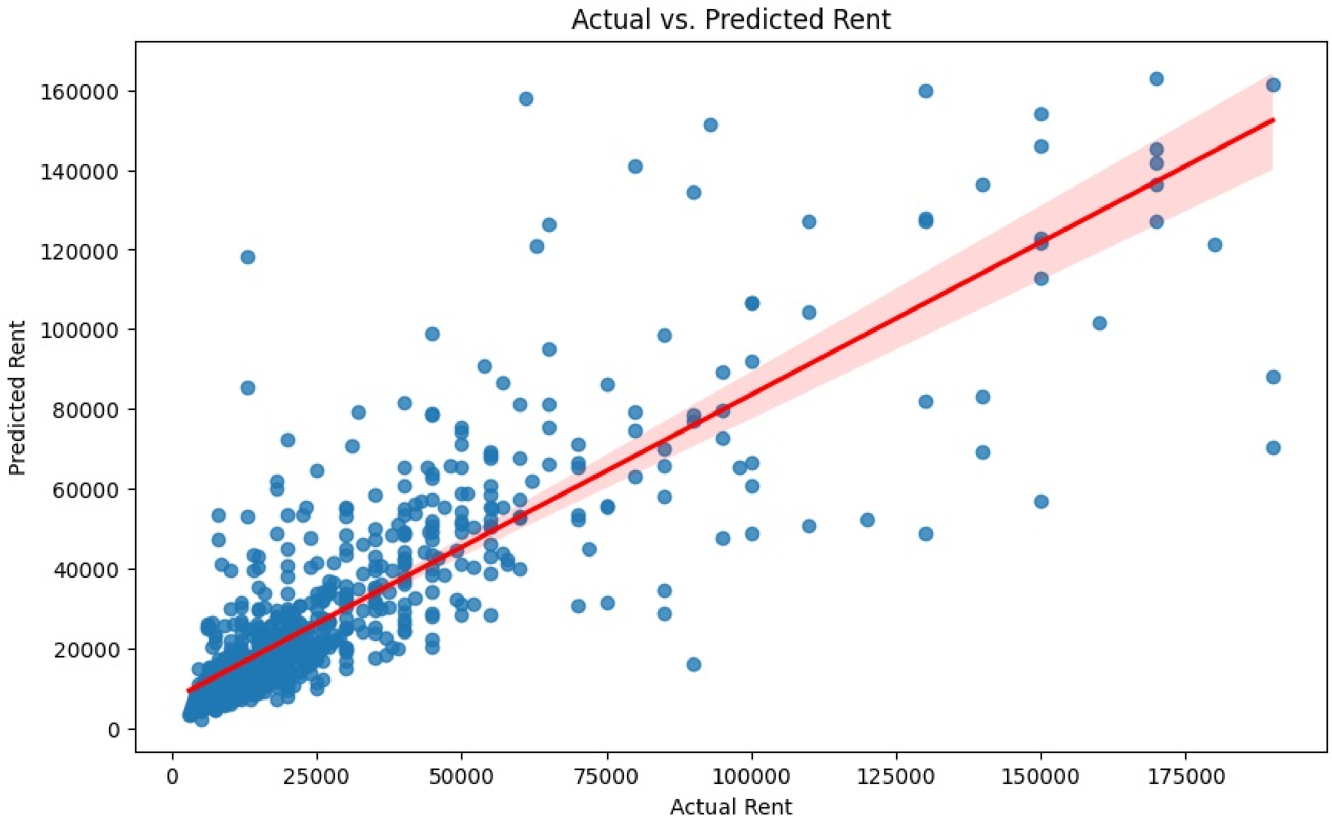
In [40]:
```
# Train Random Forest model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[40]:
```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [41]:
```
# Evaluate model
y_pred = model.predict(X_test)
print("R2 Score:", r2_score(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("MAE:", mean_absolute_error(y_test, y_pred))
```

```
R2 Score: 0.7114420037005476
RMSE: 15716.34403890847
MAE: 8354.7647195952
```

In [42]:
```
# Plot actual vs. predicted values
plt.figure(figsize=(10, 6))
sns.regplot(x=y_test, y=y_pred, line_kws={'color': 'red'})
plt.xlabel("Actual Rent")
plt.ylabel("Predicted Rent")
plt.title("Actual vs. Predicted Rent")
plt.show()
```



```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js