In [42]:
```
import pandas as pd
import numpy as np
from datetime import datetime
from scipy import stats
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [43]:
```
#1. Generate descriptive statistics for the dataset, and comment on the main trends
df = pd.read_excel('EDA.xlsx')
df.head()
```

Out[43]:

	SystemCodeNumber	Capacity	Occupancy	per_usage	per_occupancy	year	month	day	Work
0	BHMBCCMKT01	577	61.0	10.57	0 - 25	2016	Oct	Tue	
1	BHMBCCMKT01	577	64.0	11.09	0 - 25	2016	Oct	Tue	
2	BHMBCCMKT01	577	80.0	13.86	0 - 25	2016	Oct	Tue	
3	BHMBCCMKT01	577	107.0	18.54	0 - 25	2016	Oct	Tue	
4	BHMBCCMKT01	577	150.0	26.00	25 - 50	2016	Oct	Tue	

In [44]:
```
df.describe()
df.shape
```

Out[44]:
```
(35332, 11)
```

In [4]:
```
missing_values = df.isnull().sum()
print("Missing values:\n", missing_values)
```

```
Missing values:
 SystemCodeNumber     0
Capacity              0
Occupancy            19
per_usage             7
per_occupancy        19
year                  0
month                 0
day                   1
WorkingDay            3
hour                  0
period                1
dtype: int64
```

In [5]:
```
df_dropped = df.dropna()
df_dropped.shape
```

Out[5]:
```
(35300, 11)
```

In [6]:
```
df_dropped
```

Out[6]:

	SystemCodeNumber	Capacity	Occupancy	per_usage	per_occupancy	year	month	day
0	BHMBCCMKT01	577	61.0	10.57	0 - 25	2016	Oct	Tue
1	BHMBCCMKT01	577	64.0	11.09	0 - 25	2016	Oct	Tue
2	BHMBCCMKT01	577	80.0	13.86	0 - 25	2016	Oct	Tue
3	BHMBCCMKT01	577	107.0	18.54	0 - 25	2016	Oct	Tue
4	BHMBCCMKT01	577	150.0	26.00	25 - 50	2016	Oct	Tue
...	...	...	...	...	...	...	...	...
35327	Shopping	1920	1517.0	79.01	75-100	2016	Dec	Mon
35328	Shopping	1920	1487.0	77.45	75-100	2016	Dec	Mon
35329	Shopping	1920	1432.0	74.58	50 - 75	2016	Dec	Mon
35330	Shopping	1920	1321.0	68.80	50 - 75	2016	Dec	Mon
35331	Shopping	1920	1180.0	61.46	50 - 75	2016	Dec	Mon

35300 rows × 11 columns

In [6]:
```
df_filled = df.fillna(0)
df_filled
```

Out[6]:

	SystemCodeNumber	Capacity	Occupancy	per_usage	per_occupancy	year	month	day
0	BHMBCCMKT01	577	61.0	10.57	0 - 25	2016	Oct	Tue
1	BHMBCCMKT01	577	64.0	11.09	0 - 25	2016	Oct	Tue
2	BHMBCCMKT01	577	80.0	13.86	0 - 25	2016	Oct	Tue
3	BHMBCCMKT01	577	107.0	18.54	0 - 25	2016	Oct	Tue
4	BHMBCCMKT01	577	150.0	26.00	25 - 50	2016	Oct	Tue
...	...	...	...	...	...	...	...	...
35327	Shopping	1920	1517.0	79.01	75-100	2016	Dec	Mon
35328	Shopping	1920	1487.0	77.45	75-100	2016	Dec	Mon
35329	Shopping	1920	1432.0	74.58	50 - 75	2016	Dec	Mon
35330	Shopping	1920	1321.0	68.80	50 - 75	2016	Dec	Mon
35331	Shopping	1920	1180.0	61.46	50 - 75	2016	Dec	Mon

35332 rows × 11 columns

In [7]:
```
df['Occupancy'] = df['Occupancy'].fillna(df['Occupancy'].mean())
df['Occupancy']
```

Out[7]:
```
0          61.0
1          64.0
2          80.0
3         107.0
4         150.0
          ...
35327    1517.0
35328    1487.0
35329    1432.0
35330    1321.0
35331    1180.0
Name: Occupancy, Length: 35332, dtype: float64
```

In [8]:
```
df['per_usage'] = df['per_usage'].fillna(df['per_usage'].mean())
df['per_usage']
```

Out[8]:
```
0        10.57
1        11.09
2        13.86
3        18.54
4        26.00
         ...
35327    79.01
35328    77.45
35329    74.58
35330    68.80
35331    61.46
Name: per_usage, Length: 35332, dtype: float64
```

In [9]:
```
df['per_occupancy'] = df['per_occupancy'].astype('category')
per_occupancy_mode = df['per_occupancy'].mode()[0]
df['per_occupancy'].fillna(per_occupancy_mode, inplace=True)
df['per_occupancy']
```

Out[9]:
```
0         0 - 25
1         0 - 25
2         0 - 25
3         0 - 25
4        25 - 50
          ...
35327     75-100
35328     75-100
35329    50 - 75
35330    50 - 75
35331    50 - 75
Name: per_occupancy, Length: 35332, dtype: category
Categories (4, object): ['0 - 25', '25 - 50', '50 - 75', '75-100']
```

In [10]:
```
df['day'] = df['day'].astype('category')
day_mode = df['day'].mode()[0]
df['day'].fillna(day_mode, inplace=True)
df['day']
```

Out[10]:
```
0        Tue
1        Tue
2        Tue
3        Tue
4        Tue
        ...
35327    Mon
35328    Mon
35329    Mon
35330    Mon
35331    Mon
Name: day, Length: 35332, dtype: category
Categories (7, object): ['Fri', 'Mon', 'Sat', 'Sun', 'Thu', 'Tue', 'Wed']
```

In [12]:
```
df['WorkingDay'] = df['WorkingDay'].astype('category')
WorkingDay_mode = df['WorkingDay'].mode()[0]
df['WorkingDay'].fillna(WorkingDay_mode, inplace=True)
df['WorkingDay']
```

Out[12]:
```
0        Yes
1        Yes
2        Yes
3        Yes
4        Yes
        ...
35327    Yes
35328    Yes
35329    Yes
35330    Yes
35331    Yes
Name: WorkingDay, Length: 35332, dtype: category
Categories (2, object): ['No', 'Yes']
```

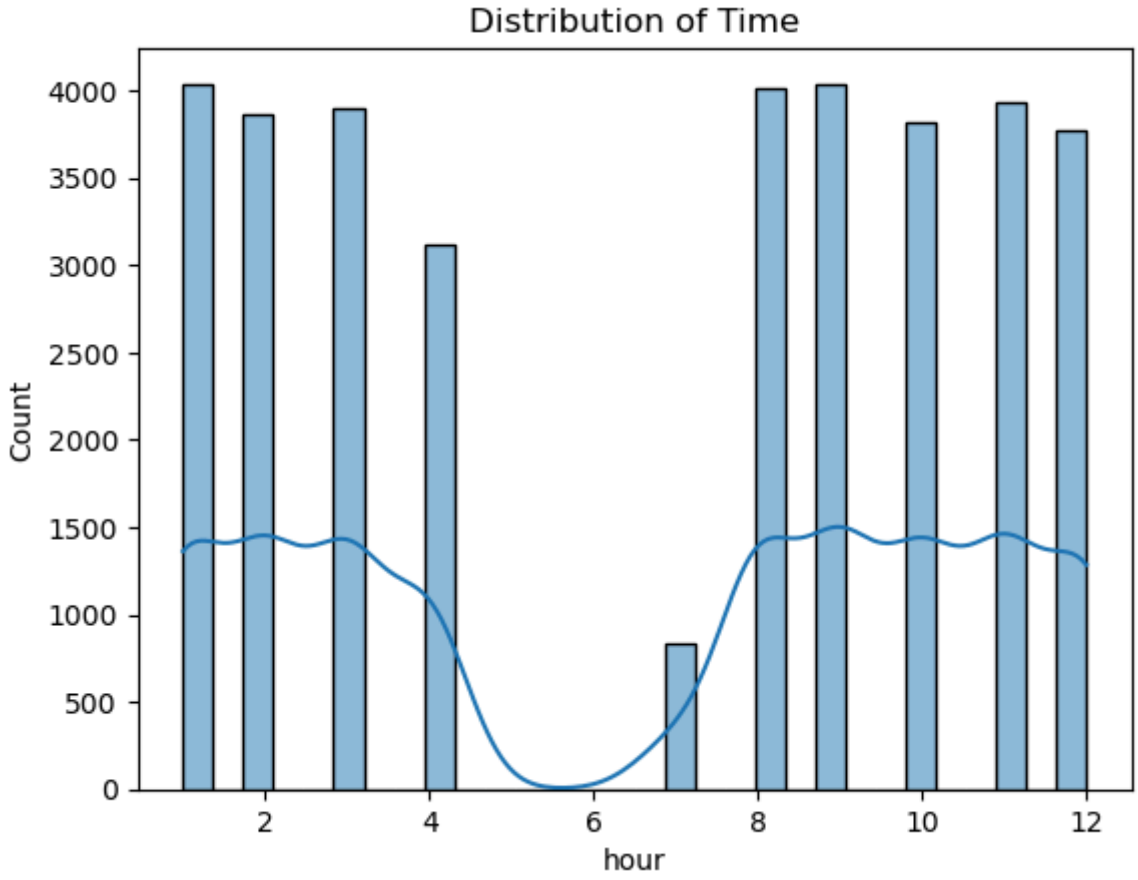
In [11]:
```
df['period'] = df['period'].astype('category')
period_mode = df['period'].mode()[0]
df['period'].fillna(period_mode, inplace=True)
df['period']
```

Out[11]:
```
0        AM
1        AM
2        AM
3        AM
4        AM
         ..
35327    PM
35328    PM
35329    PM
35330    PM
35331    PM
Name: period, Length: 35332, dtype: category
Categories (2, object): ['AM', 'PM']
```

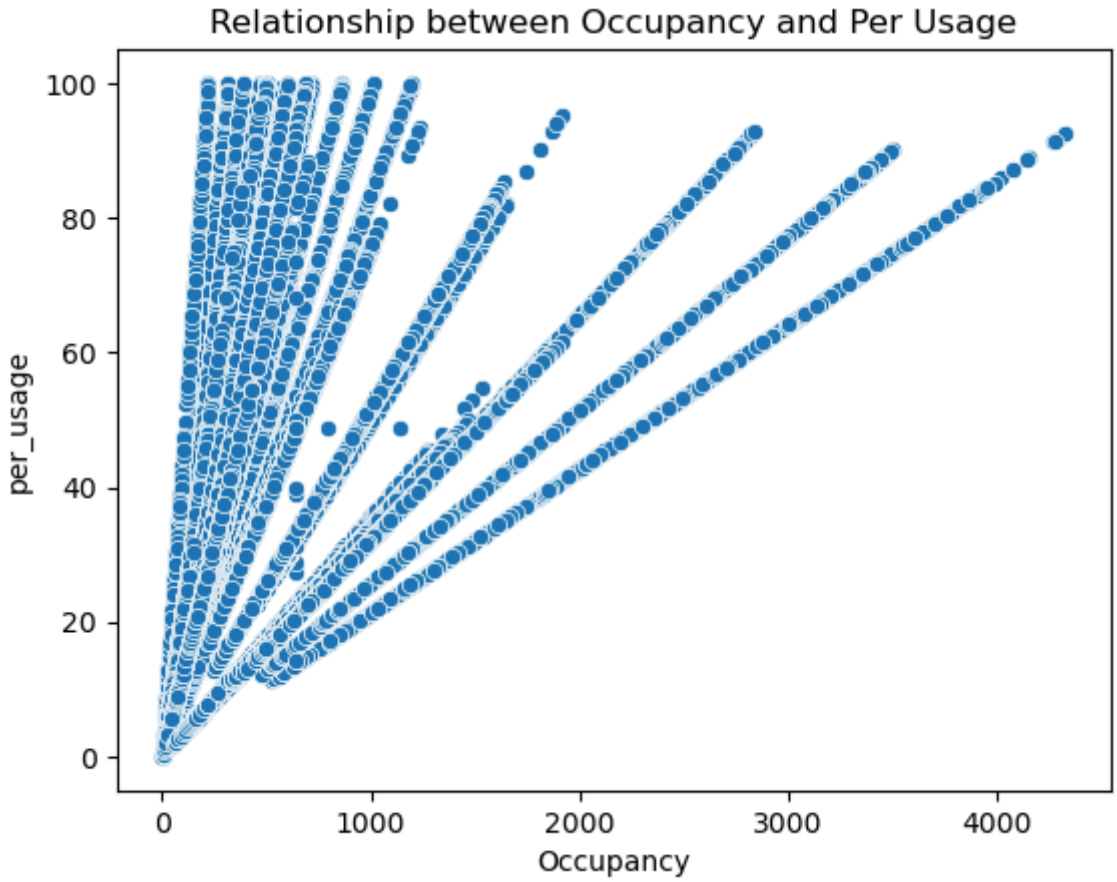
In [12]:
```
filled_values = df_filled.isnull().sum()
print("Filled values:\n", filled_values)
```

```
Filled values:
 SystemCodeNumber    0
Capacity             0
Occupancy            0
per_usage            0
per_occupancy        0
year                 0
month                0
day                  0
WorkingDay           0
hour                 0
period               0
dtype: int64
```

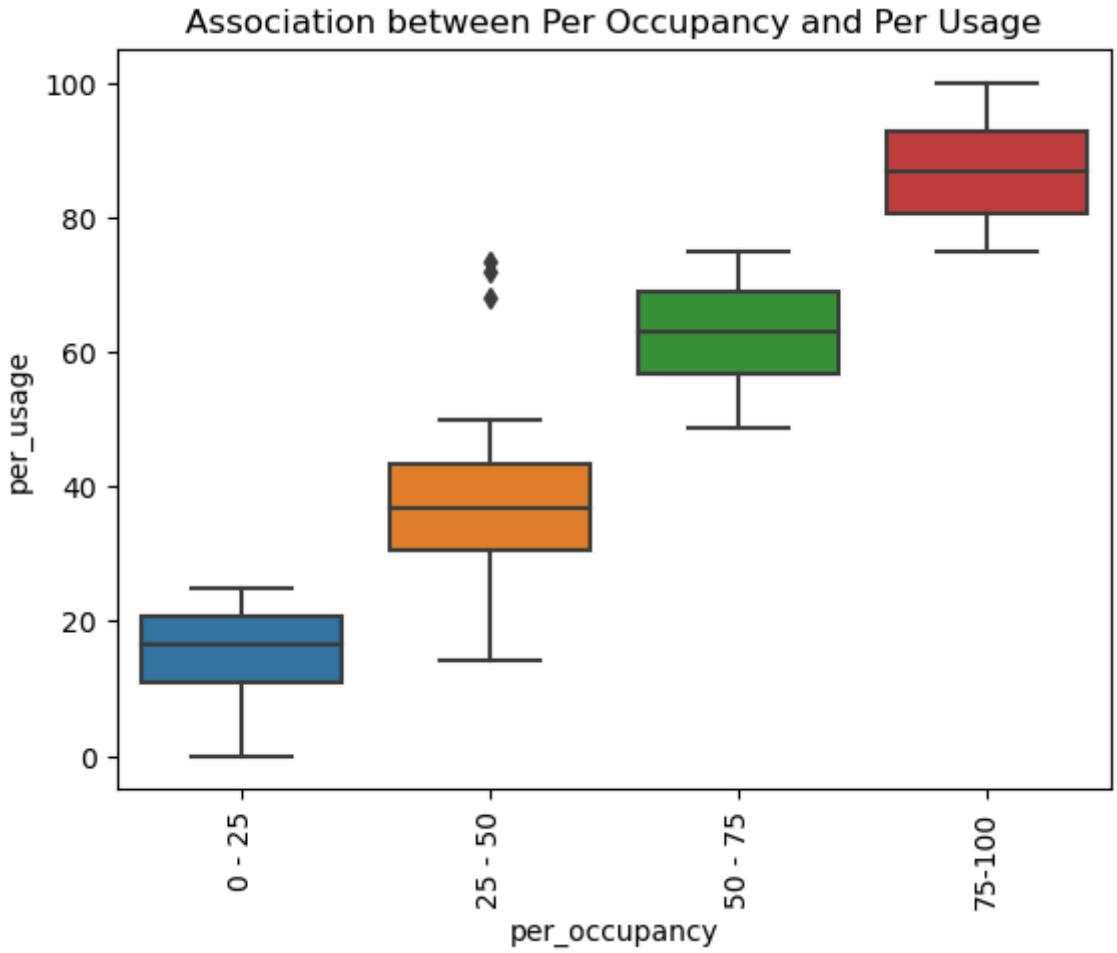
In [13]:
```
# A. Distribution of individual continuous variables
sns.histplot(data=df, x='hour', bins=30, kde=True)
plt.title('Distribution of Time')
plt.show()
```



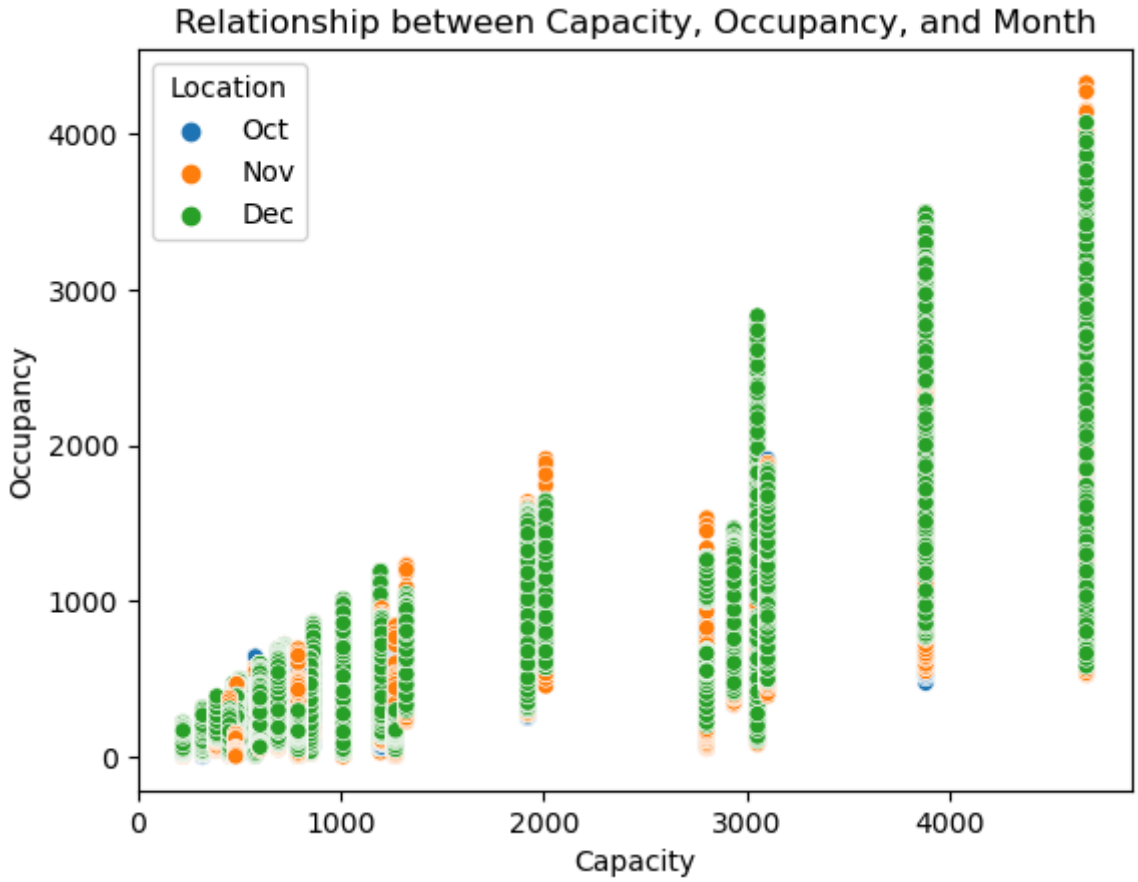
In [14]:
```
# B. Relationship between a pair of continuous variables
sns.scatterplot(data=df, x='Occupancy', y='per_usage')
plt.title('Relationship between Occupancy and Per Usage')
plt.show()
```



In [15]:
```
# C. Association between a categorical variable and a continuous one
sns.boxplot(data=df, x='per_occupancy', y='per_usage')
plt.title('Association between Per Occupancy and Per Usage')
plt.xticks(rotation=90)
plt.show()
```



In [16]:
```
# D. Relationship between more than two variables
sns.scatterplot(data=df, x='Capacity', y='Occupancy', hue='month')
plt.title('Relationship between Capacity, Occupancy, and Month')
plt.legend(title='Location')
plt.show()
```



In [41]:
```
# 4. Display unique values of a categorical variable and their frequencies.
day_counts = df['day'].value_counts()
print("Unique values of day and their counts:")
print(day_counts)
```

```
Unique values of day and their counts:
day
Tue    5425
Wed    5417
Mon    5358
Thu    4938
Fri    4920
Sun    4687
Sat    4587
Name: count, dtype: int64
```

In [18]:
```
month_counts = df['month'].value_counts()
print(month_counts)
```

```
month
Nov    14860
Oct    12432
Dec     8040
Name: count, dtype: int64
```

In [19]:
```
# 5. (i) Build a contingency table of two potentially related categorical variables
cont_table = pd.crosstab(df['day'],df['SystemCodeNumber'])
cont_table
```

Out[19]:

SystemCodeNumber	BHMBCCMKT01	BHMBCCPST01	BHMBCCSNH01	BHMBCCTHL01	BHMBRCBF
day					
Fri	180	180	180	148	
Mon	198	198	178	171	
Sat	180	178	180	138	
Sun	180	180	180	140	
Thu	180	180	180	140	
Tue	197	179	189	175	
Wed	197	179	196	160	

7 rows × 30 columns

In [20]:
```
# 5. (ii) Conduct a statistical test of the independence between them and interpret
chi2, p, dof, expected = stats.chi2_contingency(cont_table)
print(f"Chi-square test : {chi2} , p-value: {p} , degree of freedom : {dof}")
```

```
Chi-square test : 396.78707797739077 , p-value: 1.9309595429229173e-19 , degree of
freedom : 174
```

In [21]:
```
# Interpret results
alpha = 0.05  # Significance level
if p < alpha:
    print("Reject the null hypothesis. There is a significant association between d
else:
    print("Fail to reject the null hypothesis. There is no significant association
```

```
Reject the null hypothesis. There is a significant association between day and Car
parking slots.
```

In [35]:
```
# 6. Retrieve one or more subset of rows based on two or more criteria and present

# Subset based on multiple criteria
subset = df[(df["SystemCodeNumber"] == "BHMBCCPST01") & (df["hour"] > 9)]
subset.head()
```

Out[35]:

	SystemCodeNumber	Capacity	Occupancy	per_usage	per_occupancy	year	month	day	W
1317	BHMBCCPST01	317	142.0	44.79	25 - 50	2016	Oct	Tue	
1318	BHMBCCPST01	317	156.0	49.21	25 - 50	2016	Oct	Tue	
1319	BHMBCCPST01	317	167.0	52.68	50 - 75	2016	Oct	Tue	
1320	BHMBCCPST01	317	175.0	55.21	50 - 75	2016	Oct	Tue	
1321	BHMBCCPST01	317	191.0	60.25	50 - 75	2016	Oct	Tue	

In [36]:
```
# Descriptive Statistics
subset.describe()
```

Out[36]:

	Capacity	Occupancy	per_usage	year	hour
count	416.0	416.000000	416.000000	416.0	416.000000
mean	317.0	137.987981	43.529399	2016.0	11.002404
std	0.0	38.657375	12.194957	0.0	0.814030
min	317.0	38.000000	11.990000	2016.0	10.000000
25%	317.0	114.000000	35.960000	2016.0	10.000000
50%	317.0	137.000000	43.220000	2016.0	11.000000
75%	317.0	159.000000	50.160000	2016.0	12.000000
max	317.0	286.000000	90.220000	2016.0	12.000000

In [39]:
```
# 7. Conduct a statistical test of the significance of the difference between the m
subset1 = df[df["SystemCodeNumber"] == "BHMBCCPST01"]
subset2 = df[df["SystemCodeNumber"] == "BHMBCCSNH01"]
# Interpret the results
t_stat, p_value = stats.ttest_ind(subset1["Occupancy"], subset2["per_usage"])
print(f" T-statistics : {t_stat}, P-value : {p_value}")
```

```
 T-statistics : 37.880316564966705, P-value : 1.3499691004701775e-249
```

In [26]:
```
# 8. Create one or more tables that group the data by a certain categorical variabl
Table = df.groupby("SystemCodeNumber")['hour'].mean().reset_index()
Table
```

Out[26]:

	SystemCodeNumber	hour
0	BHMBCCMKT01	6.693598
1	BHMBCCPST01	6.693093
2	BHMBCCSNH01	6.667186
3	BHMBCCTHL01	6.983209
4	BHMBRCBRG01	6.780909
5	BHMBRCBRG02	6.775952
6	BHMBRCBRG03	6.693086
7	BHMBRTARC01	6.761364
8	BHMEURBRD01	6.696646
9	BHMEURBRD02	6.703762
10	BHMMBMMBX01	6.697941
11	BHMNCPHST01	6.694360
12	BHMNCPLDH01	6.674671
13	BHMNCPNHS01	6.733591
14	BHMNCPNST01	6.685213
15	BHMNCPPLS01	6.659954
16	BHMNCPRAN01	6.689713
17	Broad Street	6.695884
18	Bull Ring	6.693086
19	NIA Car Parks	6.696844
20	NIA North	6.900000
21	NIA South	6.696844
22	Others-CCCPS105a	6.693598
23	Others-CCCPS119a	6.693598
24	Others-CCCPS133	6.695518
25	Others-CCCPS135a	6.693598
26	Others-CCCPS202	6.693598
27	Others-CCCPS8	6.693598
28	Others-CCCPS98	6.693598
29	Shopping	6.693598

In [27]:
```
# 9. Implement a linear regression model and interpret its output including its acc
import statsmodels.api as sm
```

In [31]:
```
# Linear regression model
model = sm.OLS.from_formula('Capacity ~ per_usage + Occupancy', data=df).fit()
model.summary()
```

Out[31]:

OLS Regression Results

Dep. Variable:	Capacity	**R-squared:**	0.799
Model:	OLS	**Adj. R-squared:**	0.799
Method:	Least Squares	**F-statistic:**	7.043e+04
Date:	Tue, 26 Mar 2024	**Prob (F-statistic):**	0.00
Time:	14:04:48	**Log-Likelihood:**	-2.7173e+05
No. Observations:	35332	**AIC:**	5.435e+05
Df Residuals:	35329	**BIC:**	5.435e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1338.6284	5.947	225.111	0.000	1326.973	1350.284
per_usage	-20.9304	0.113	-185.615	0.000	-21.151	-20.709
Occupancy	1.6942	0.005	370.837	0.000	1.685	1.703

Omnibus:	6867.426	**Durbin-Watson:**	0.114
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	17342.899
Skew:	1.072	**Prob(JB):**	0.00
Kurtosis:	5.680	**Cond. No.**	1.95e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.95e+03. This might indicate that there are strong multicollinearity or other numerical problems.

In [40]:
```
# Interpret the results
print("\nInterpretation:")
print("\nThe R-squared value is:", model.rsquared)
```

```
Interpretation:

The R-squared value is: 0.7994743496015648
```

In [ ]: