Atliq Grands Data Analysis

Atliq Grands Company is the hotel chain which operates in various cities in India namely Delhi, Mumbai, Hyd and bang. In industry they are since 20 yrs. They have different types of hotels such as Atliq Season, Exotica, Bay and Palace and thay have different types of rooms such as Std, Elite, Premium and Presedential. Hotel booking can be done through variety of mediums : their website,the other third party websites such as make your trip, log trip, etc. and offline. The booking data collected goes to the booking db of the Atliq Grands.

Problem:

Atliq is facing a major problem from their competitors. They are loosing the revenue and the market share and hence Atliq mgt has decided to onboard the Data Analyst team to resolve this issue so to do informed decision making which will increase their revenue.

In [ ]:
```
import pandas as pd
```

In [2]:
```
df_bookings = pd.read_csv("C:\\Users\\Aditi\\Downloads\\datasets\\fact_bookings.csv")
```

In [3]:
```
df_bookings.head()
```

Out[3]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	1.0
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	5.0
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	NaN
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0

In [4]:
```
df_bookings.shape
```

Out[4]:
```
(134590, 12)
```

In [5]:
```
df_bookings.room_category.unique()
```

Out[5]:
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

In [6]:
```
df_bookings.booking_platform.unique()
```

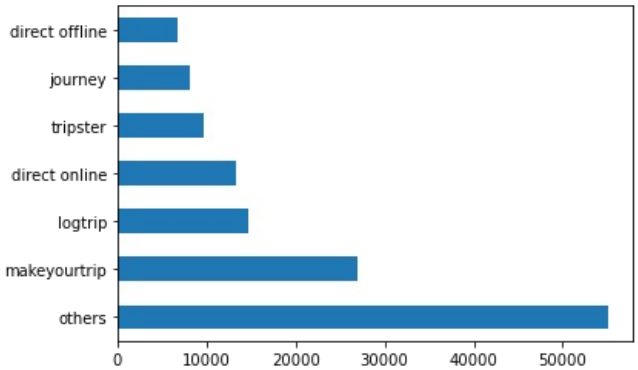
Out[6]:
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

In [7]:
```
# Count of bookings per platform :

df_bookings.booking_platform.value_counts()
```

Out[7]:
```
others            55066
makeyourtrip      26898
logtrip           14756
direct online     13379
tripster           9630
journey            8106
direct offline     6755
Name: booking_platform, dtype: int64
```

In [8]:
```
# Bar chart of the bookings per platform:

df_bookings.booking_platform.value_counts().plot(kind="barh")
```

Out[8]:
```
<AxesSubplot:>
```



In [9]:
```
df_bookings.describe()
```

Out[9]:

	property_id	no_guests	ratings_given	revenue_generated	revenue_realized
count	134590.000000	134587.000000	56683.000000	1.345900e+05	134590.000000
mean	18061.113493	2.036170	3.619004	1.537805e+04	12696.123256
std	1093.055847	1.034885	1.235009	9.303604e+04	6928.108124
min	16558.000000	-17.000000	1.000000	6.500000e+03	2600.000000
25%	17558.000000	1.000000	3.000000	9.900000e+03	7600.000000
50%	17564.000000	2.000000	4.000000	1.350000e+04	11700.000000
75%	18563.000000	2.000000	5.000000	1.800000e+04	15300.000000
max	19563.000000	6.000000	5.000000	2.856000e+07	45220.000000

In [10]:
```
df_bookings.revenue_generated.min(),df_bookings.revenue_generated.max()
```

Out[10]:
```
(6500, 28560000)
```

In [11]:
```
df_date = pd.read_csv("C:\\Users\\Aditi\\Downloads\\datasets\\dim_date.csv")
df_hotels=pd.read_csv("C:\\Users\\Aditi\\Downloads\\datasets\\dim_hotels.csv")
df_rooms=pd.read_csv("C:\\Users\\Aditi\\Downloads\\datasets\\dim_rooms.csv")
df_agg_bookings=pd.read_csv("C:\\Users\\Aditi\\Downloads\\datasets\\fact_aggregated_bookings.csv")
```

In [12]:
```
df_hotels.shape
```

Out[12]:
```
(25, 4)
```

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

Out[13]:

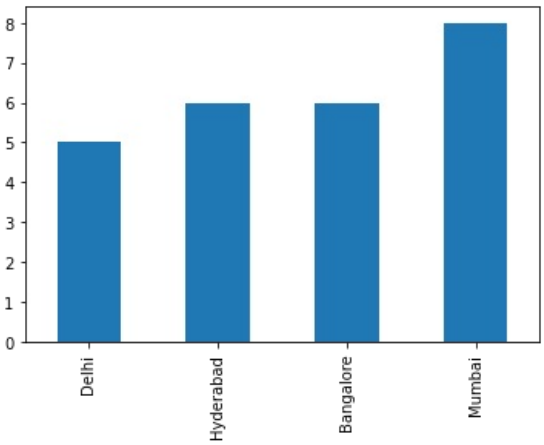
	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi
3	16561	Atliq Blu	Luxury	Delhi
4	16562	Atliq Bay	Luxury	Delhi

In [14]:
```
df_hotels.category.value_counts()
```

Out[14]:
```
Luxury      16
Business     9
Name: category, dtype: int64
```

In [15]:
```
# How many hotels does every city has:

df_hotels.city.value_counts().sort_values().plot(kind="bar")
```

Out[15]:
```
<AxesSubplot:>
```



In [16]:
```
# Exploration of agg. hotel bookings:

df_agg_bookings.head()
```

Out[16]:

	property_id	check_in_date	room_category	successful_bookings	capacity
0	16559	1-May-22	RT1	25	30.0
1	19562	1-May-22	RT1	28	30.0
2	19563	1-May-22	RT1	23	30.0
3	17558	1-May-22	RT1	30	19.0
4	16558	1-May-22	RT1	18	19.0

In [17]:
```
# Unique property ids:

df_agg_bookings.property_id.nunique()
```

Out[17]:
```
25
```

In [18]:
```
# Total bookings per property id:

result = df_agg_bookings.successful_bookings.value_counts().sort_values(ascending=False)
result
```

Out[18]:
```
12     545
14     534
10     511
11     510
9      482
13     479
15     464
3      415
17     406
18     391
8      384
16     362
19     345
20     337
22     267
21     267
23     251
4      243
7      232
2      225
6      191
5      186
24     169
25     159
26     148
28     131
27      99
29      70
31      58
30      49
34      46
33      45
32      44
35      33
1       28
36      28
38      23
37      20
40       9
39       8
43       2
100      1
50       1
41       1
123      1
Name: successful_bookings, dtype: int64
```

In [19]:
```
# Find out the days at which bookings are greater than capacity:

overbooked_days = df_agg_bookings[df_agg_bookings['successful_bookings'] > df_agg_bookings['capacity']]

# Display the overbooked days
print(overbooked_days)
```

```
      property_id check_in_date room_category  successful_bookings  capacity
3           17558      1-May-22           RT1                   30      19.0
12          16563      1-May-22           RT1                  100      41.0
4136        19558     11-Jun-22           RT2                   50      39.0
6209        19560      2-Jul-22           RT1                  123      26.0
8522        19559     25-Jul-22           RT1                   35      24.0
9194        18563     31-Jul-22           RT4                   20      18.0
```

In [20]:
```
# Find out properties that have highest capacity:

# Sort the DataFrame by 'Capacity' in descending order
sorted_df = df_agg_bookings.sort_values(by='capacity', ascending=False)
```

```
# Get the properties with the highest capacity
highest_capacity_properties = sorted_df[sorted_df['capacity'] == sorted_df['capacity'].max()]

# Display the properties with the highest capacity
print(highest_capacity_properties)
```

```
      property_id check_in_date room_category  successful_bookings  capacity
3128        17558      1-Jun-22           RT2                   19      50.0
2128        17558     22-May-22           RT2                   38      50.0
1728        17558     18-May-22           RT2                   21      50.0
5828        17558     28-Jun-22           RT2                   26      50.0
3928        17558      9-Jun-22           RT2                   27      50.0
...           ...           ...           ...                  ...       ...
8528        17558     25-Jul-22           RT2                   23      50.0
4128        17558     11-Jun-22           RT2                   36      50.0
628         17558      7-May-22           RT2                   39      50.0
5027        17558     20-Jun-22           RT2                   21      50.0
328         17558      4-May-22           RT2                   27      50.0

[92 rows x 5 columns]
```

Data Cleaning

In [21]:
```
df_bookings.describe()
```

Out[21]:

	property_id	no_guests	ratings_given	revenue_generated	revenue_realized
count	134590.000000	134587.000000	56683.000000	1.345900e+05	134590.000000
mean	18061.113493	2.036170	3.619004	1.537805e+04	12696.123256
std	1093.055847	1.034885	1.235009	9.303604e+04	6928.108124
min	16558.000000	-17.000000	1.000000	6.500000e+03	2600.000000
25%	17558.000000	1.000000	3.000000	9.900000e+03	7600.000000
50%	17564.000000	2.000000	4.000000	1.350000e+04	11700.000000
75%	18563.000000	2.000000	5.000000	1.800000e+04	15300.000000
max	19563.000000	6.000000	5.000000	2.856000e+07	45220.000000

In [22]:
```
df_bookings[df_bookings.no_guests<=0]
```

Out[22]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	
17924	May122218559RT44	18559	12/5/2022	12/5/2022	14-05-22	-10.0	RT4	direct online	
18020	May122218561RT22	18561	8/5/2022	12/5/2022	14-05-22	-12.0	RT2	makeyourtrip	
18119	May122218562RT311	18562	5/5/2022	12/5/2022	17-05-22	-6.0	RT3	direct offline	
18121	May122218562RT313	18562	10/5/2022	12/5/2022	17-05-22	-4.0	RT3	direct online	
56715	Jun082218562RT12	18562	5/6/2022	8/6/2022	13-06-22	-17.0	RT1	others	
119765	Jul202219560RT220	19560	19-07-22	20-07-22	22-07-22	-1.0	RT2	others	
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	-4.0	RT4	logtrip	

In [23]:
```
df_bookings = df_bookings[df_bookings.no_guests>0]
df_bookings
```

Out[23]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_g
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	
...	...	...	...	...	...	...	...	...	
134584	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	2.0	RT4	others	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	1.0	RT4	makeyourtrip	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	1.0	RT4	tripster	
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	2.0	RT4	logtrip	
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	2.0	RT4	makeyourtrip	

134578 rows × 12 columns

In [24]:
```
# Removal of outliers:

avg,std = df_bookings.revenue_generated.mean(),df_bookings.revenue_generated.std()
```

In [25]:
```
avg,std
```

Out[25]:
```
(15378.036937686695, 93040.15493143328)
```

In [26]:
```
higher_limit = avg + 3*std
higher_limit
```

Out[26]:
```
294498.50173198653
```

In [27]:
```
lower_limit = avg - 3*std
lower_limit
```

Out[27]:
```
-263742.4278566132
```

In [28]:
```
# Outliers:

df_bookings[df_bookings.revenue_generated > higher_limit]
```

Out[28]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	
111	May012216559RT32	16559	29-04-22	1/5/2022	2/5/2022	6.0	RT3	direct online	
315	May012216562RT22	16562	28-04-22	1/5/2022	4/5/2022	2.0	RT2	direct offline	
562	May012217559RT118	17559	26-04-22	1/5/2022	2/5/2022	2.0	RT1	others	
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	2.0	RT2	direct online	

In [29]:
```
# Accepted Values:


df_bookings = df_bookings[df_bookings.revenue_generated < higher_limit]
df_bookings
```

Out[29]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_g
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	
...	...	...	...	...	...	...	...	...	
134584	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	2.0	RT4	others	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	1.0	RT4	makeyourtrip	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	1.0	RT4	tripster	
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	2.0	RT4	logtrip	
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	2.0	RT4	makeyourtrip	

134573 rows × 12 columns

In [30]:
```
df_bookings.shape
```

Out[30]:
```
(134573, 12)
```

In [31]:
```
# Analysing revenue realised column:

df_bookings.revenue_realized.describe()
```

Out[31]:
```
count    134573.000000
mean      12695.983585
std        6927.791692
min        2600.000000
25%        7600.000000
50%       11700.000000
75%       15300.000000
max       45220.000000
Name: revenue_realized, dtype: float64
```

Data Transformation

In [32]:
```
df_agg_bookings.head()
```

Out[32]:

	property_id	check_in_date	room_category	successful_bookings	capacity
0	16559	1-May-22	RT1	25	30.0
1	19562	1-May-22	RT1	28	30.0
2	19563	1-May-22	RT1	23	30.0
3	17558	1-May-22	RT1	30	19.0
4	16558	1-May-22	RT1	18	19.0

In [33]:
```
## occupancy percentage

df_agg_bookings["occ_percentage"] = df_agg_bookings["successful_bookings"]/df_agg_bookings["capacity"]
```

In [34]:
```
df_agg_bookings.head()
```

Out[34]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_percentage
0	16559	1-May-22	RT1	25	30.0	0.833333
1	19562	1-May-22	RT1	28	30.0	0.933333
2	19563	1-May-22	RT1	23	30.0	0.766667
3	17558	1-May-22	RT1	30	19.0	1.578947
4	16558	1-May-22	RT1	18	19.0	0.947368

In [35]:
```
df_agg_bookings["occ_percentage"] = df_agg_bookings["occ_percentage"].apply(lambda x : round(x*100,2))
```

In [36]:
```
df_agg_bookings.head()
```

Out[36]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_percentage
0	16559	1-May-22	RT1	25	30.0	83.33
1	19562	1-May-22	RT1	28	30.0	93.33
2	19563	1-May-22	RT1	23	30.0	76.67
3	17558	1-May-22	RT1	30	19.0	157.89
4	16558	1-May-22	RT1	18	19.0	94.74

Insights Generation

In [37]:
```
# What is the avg occupancy rate in each of the room categories:

df_agg_bookings.groupby("room_category")["occ_percentage"].mean().round(2)
```

Out[37]:
```
room_category
RT1    58.22
RT2    58.04
RT3    58.03
RT4    59.30
Name: occ_percentage, dtype: float64
```

In [38]:
```
df_rooms
```

Out[38]:

	room_id	room_class
0	RT1	Standard
1	RT2	Elite
2	RT3	Premium
3	RT4	Presidential

In [39]:
```
df = pd.merge(df_agg_bookings, df_rooms, left_on = "room_category", right_on="room_id" )
df.head()
```

Out[39]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_percentage	room_id	room_class
0	16559	1-May-22	RT1	25	30.0	83.33	RT1	Standard
1	19562	1-May-22	RT1	28	30.0	93.33	RT1	Standard
2	19563	1-May-22	RT1	23	30.0	76.67	RT1	Standard
3	17558	1-May-22	RT1	30	19.0	157.89	RT1	Standard
4	16558	1-May-22	RT1	18	19.0	94.74	RT1	Standard

In [40]:
```
df.groupby("room_class")["occ_percentage"].mean().round(2)
```

Out[40]:
```
room_class
Elite           58.04
Premium         58.03
Presidential    59.30
Standard        58.22
Name: occ_percentage, dtype: float64
```

In [41]:
```
df.drop("room_id", axis=1, inplace=True)
df.head()
```

Out[41]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_percentage	room_class
0	16559	1-May-22	RT1	25	30.0	83.33	Standard
1	19562	1-May-22	RT1	28	30.0	93.33	Standard
2	19563	1-May-22	RT1	23	30.0	76.67	Standard
3	17558	1-May-22	RT1	30	19.0	157.89	Standard
4	16558	1-May-22	RT1	18	19.0	94.74	Standard

In [42]:
```
## Print average occ percentage per city:

df_hotels.head()
```

Out[42]:

	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi
3	16561	Atliq Blu	Luxury	Delhi
4	16562	Atliq Bay	Luxury	Delhi

In [43]:
```
city_avg_occ_pct = pd.merge(df, df_hotels, on="property_id")
```

In [44]:
```
city_avg_occ_pct
```

Out[44]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_percentage	room_class	property_name	category	
0	16559	1-May-22	RT1	25	30.0	83.33	Standard	Atliq Exotica	Luxury	M
1	16559	2-May-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury	M
2	16559	3-May-22	RT1	17	30.0	56.67	Standard	Atliq Exotica	Luxury	M
3	16559	4-May-22	RT1	21	30.0	70.00	Standard	Atliq Exotica	Luxury	M
4	16559	5-May-22	RT1	16	30.0	53.33	Standard	Atliq Exotica	Luxury	M
...	...	...	...	...	...	...	...	...	...	
9195	18560	27-Jul-22	RT4	6	15.0	40.00	Presidential	Atliq City	Business	Hyd
9196	18560	28-Jul-22	RT4	9	15.0	60.00	Presidential	Atliq City	Business	Hyd
9197	18560	29-Jul-22	RT4	8	15.0	53.33	Presidential	Atliq City	Business	Hyd
9198	18560	30-Jul-22	RT4	9	15.0	60.00	Presidential	Atliq City	Business	Hyd
9199	18560	31-Jul-22	RT4	12	15.0	80.00	Presidential	Atliq City	Business	Hyd

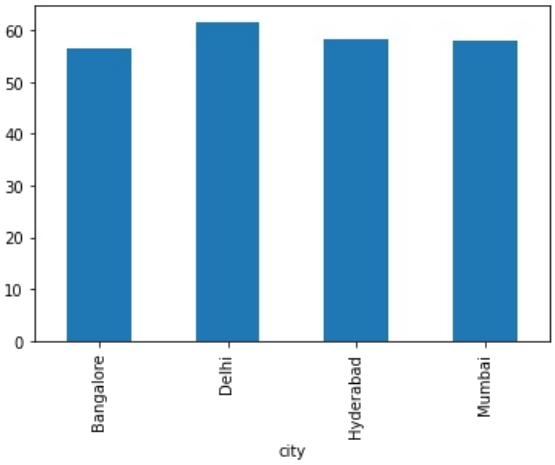
9200 rows × 10 columns

In [45]:
```
city_avg_occ_pct.groupby("city")["occ_percentage"].mean()
```

Out[45]:
```
city
Bangalore    56.594207
Delhi        61.606467
Hyderabad    58.144651
Mumbai       57.936305
Name: occ_percentage, dtype: float64
```

In [46]:
```
# Plotting the result:

city_avg_occ_pct.groupby("city")["occ_percentage"].mean().plot(kind="bar")
```

Out[46]:
```
<AxesSubplot:xlabel='city'>
```



In [47]:
```
## When was the occupancy better : Weekday or weekend?

df = pd.merge(df, df_date, left_on = "check_in_date", right_on="date" )
df.head()
```

Out[47]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_percentage	room_class	date	mmm yy	week no	day_type
0	18560	10-May-22	RT1	19	30.0	63.33	Standard	10-May-22	May 22	W 20	weekeday
1	19562	10-May-22	RT1	18	30.0	60.00	Standard	10-May-22	May 22	W 20	weekeday
2	19563	10-May-22	RT1	16	30.0	53.33	Standard	10-May-22	May 22	W 20	weekeday
3	17558	10-May-22	RT1	11	19.0	57.89	Standard	10-May-22	May 22	W 20	weekeday
4	16558	10-May-22	RT1	10	19.0	52.63	Standard	10-May-22	May 22	W 20	weekeday

In [48]:
```
df.groupby("day_type")["occ_percentage"].mean().round(2)
```

Out[48]:
```
day_type
weekeday    50.90
weekend     72.39
Name: occ_percentage, dtype: float64
```

In [49]:
```
## For the month of June what was the occupancy rate for different cities:

df["mmm yy"].unique()
```

Out[49]:
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

In [50]:
```
df_jun_22 = df[df["mmm yy"]=="Jun 22"]
df_jun_22
```

Out[50]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_percentage	room_class	date	mmm yy	week no	day_type
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	10-Jun-22	Jun 22	W 24	weekeday
2201	19562	10-Jun-22	RT1	19	30.0	63.33	Standard	10-Jun-22	Jun 22	W 24	weekeday
2202	19563	10-Jun-22	RT1	17	30.0	56.67	Standard	10-Jun-22	Jun 22	W 24	weekeday
2203	17558	10-Jun-22	RT1	9	19.0	47.37	Standard	10-Jun-22	Jun 22	W 24	weekeday
2204	16558	10-Jun-22	RT1	11	19.0	57.89	Standard	10-Jun-22	Jun 22	W 24	weekeday
...	...	...	...	...	...	...	...	...	...	...	...
4295	17562	30-Jun-22	RT4	3	6.0	50.00	Presidential	30-Jun-22	Jun 22	W 27	weekeday
4296	19563	30-Jun-22	RT4	3	6.0	50.00	Presidential	30-Jun-22	Jun 22	W 27	weekeday
4297	16560	30-Jun-22	RT4	3	7.0	42.86	Presidential	30-Jun-22	Jun 22	W 27	weekeday
4298	19558	30-Jun-22	RT4	3	7.0	42.86	Presidential	30-Jun-22	Jun 22	W 27	weekeday
4299	17561	30-Jun-22	RT4	3	4.0	75.00	Presidential	30-Jun-22	Jun 22	W 27	weekeday

2100 rows × 11 columns

In [51]:
```
df1 = pd.merge(df_jun_22,df_hotels,on="property_id")
df1
```

Out[51]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_percentage	room_class	date	mmm yy	week no	day_type
0	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	10-Jun-22	Jun 22	W 24	weekeday
1	16559	10-Jun-22	RT2	26	41.0	63.41	Elite	10-Jun-22	Jun 22	W 24	weekeday
2	16559	10-Jun-22	RT3	20	32.0	62.50	Premium	10-Jun-22	Jun 22	W 24	weekeday
3	16559	10-Jun-22	RT4	11	18.0	61.11	Presidential	10-Jun-22	Jun 22	W 24	weekeday
4	16559	11-Jun-22	RT1	27	30.0	90.00	Standard	11-Jun-22	Jun 22	W 24	weekend
...	...	...	...	...	...	...	...	...	...	...	...
2095	18560	29-Jun-22	RT4	9	15.0	60.00	Presidential	29-Jun-22	Jun 22	W 27	weekeday
2096	18560	30-Jun-22	RT1	18	30.0	60.00	Standard	30-Jun-22	Jun 22	W 27	weekeday
2097	18560	30-Jun-22	RT2	24	40.0	60.00	Elite	30-Jun-22	Jun 22	W 27	weekeday
2098	18560	30-Jun-22	RT3	14	24.0	58.33	Premium	30-Jun-22	Jun 22	W 27	weekeday
2099	18560	30-Jun-22	RT4	8	15.0	53.33	Presidential	30-Jun-22	Jun 22	W 27	weekeday

2100 rows × 14 columns

In [52]:
```
df1.groupby("city")["occ_percentage"].mean().round(2).sort_values(ascending=False)
```

Out[52]:
```
city
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Bangalore    56.58
Name: occ_percentage, dtype: float64
```

In [53]:
```
df_aug = pd.read_csv("C:\\Users\\Aditi\\Downloads\\datasets\\new_data_august.csv")
df_aug.head()
```

Out[53]:

	property_id	property_name	category	city	room_category	room_class	check_in_date	mmm yy	week no	day_type	successful_bookings
0	16559	Atliq Exotica	Luxury	Mumbai	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	30
1	19562	Atliq Bay	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	21
2	19563	Atliq Palace	Business	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	23
3	19558	Atliq Grands	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	30
4	19560	Atliq City	Business	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	20

In [63]:
```
df = pd.merge(df,df_hotels,on="property_id")
df.head()
```

Out[63]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_percentage	room_class	date	mmm yy	week no	day_type	pr
0	18560	10-May-22	RT1	19	30.0	63.33	Standard	10-May-22	May 22	W 20	weekeday	
1	18560	10-May-22	RT2	25	40.0	62.50	Elite	10-May-22	May 22	W 20	weekeday	
2	18560	10-May-22	RT3	14	24.0	58.33	Premium	10-May-22	May 22	W 20	weekeday	
3	18560	10-May-22	RT4	9	15.0	60.00	Presidential	10-May-22	May 22	W 20	weekeday	
4	18560	11-May-22	RT1	20	30.0	66.67	Standard	11-May-22	May 22	W 20	weekeday	

In [64]:
```
df_aug.columns
```

Out[64]:
```
Index(['property_id', 'property_name', 'category', 'city', 'room_category',
       'room_class', 'check_in_date', 'mmm yy', 'week no', 'day_type',
       'successful_bookings', 'capacity', 'occ%'],
      dtype='object')
```

In [65]:
```
df.columns
```

Out[65]:
```
Index(['property_id', 'check_in_date', 'room_category', 'successful_bookings',
       'capacity', 'occ_percentage', 'room_class', 'date', 'mmm yy', 'week no',
       'day_type', 'property_name', 'category', 'city'],
      dtype='object')
```

In [69]:
```
latest_df = pd.concat([df,df_aug],ignore_index=True,axis=0)
latest_df.tail(10)
```

Out[69]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_percentage	room_class	date	mmm yy	week no	day_type
6497	16559	31-Jul-22	RT2	29	41.0	70.73	Elite	31-Jul-22	Jul 22	W 32	weekend
6498	16559	31-Jul-22	RT3	22	32.0	68.75	Premium	31-Jul-22	Jul 22	W 32	weekend
6499	16559	31-Jul-22	RT4	13	18.0	72.22	Presidential	31-Jul-22	Jul 22	W 32	weekend
6500	16559	01-Aug-22	RT1	30	30.0	NaN	Standard	NaN	Aug-22	W 32	weekeday
6501	19562	01-Aug-22	RT1	21	30.0	NaN	Standard	NaN	Aug-22	W 32	weekeday
6502	19563	01-Aug-22	RT1	23	30.0	NaN	Standard	NaN	Aug-22	W 32	weekeday
6503	19558	01-Aug-22	RT1	30	40.0	NaN	Standard	NaN	Aug-22	W 32	weekeday
6504	19560	01-Aug-22	RT1	20	26.0	NaN	Standard	NaN	Aug-22	W 32	weekeday
6505	17561	01-Aug-22	RT1	18	26.0	NaN	Standard	NaN	Aug-22	W 32	weekeday
6506	17564	01-Aug-22	RT1	10	16.0	NaN	Standard	NaN	Aug-22	W 32	weekeday

In [70]:
```
latest_df.shape
```

Out[70]:
```
(6507, 15)
```

In [74]:
```
## Print revenue realised per city:

df_bookings_all = pd.merge(df_bookings,df_hotels,on="property_id")
df_bookings_all
```

Out[74]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_g
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	
3	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	
4	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	
...	...	...	...	...	...	...	...	...	
134568	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	2.0	RT4	others	
134569	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	1.0	RT4	makeyourtrip	
134570	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	1.0	RT4	tripster	
134571	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	2.0	RT4	logtrip	
134572	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	2.0	RT4	makeyourtrip	

134573 rows × 15 columns

In [76]:
```
df_bookings_all.groupby("city")["revenue_realized"].sum()
```

Out[76]:
```
city
Bangalore    420383550
Delhi        294404488
Hyderabad    325179310
Mumbai       668569251
Name: revenue_realized, dtype: int64
```

In [77]:
```
# Print month by month revenue:

pd.merge(df_bookings_all, df_date, left_on="check_in_date",right_on="date")
```

Out[77]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_

In [79]:
```
df_bookings_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 134573 entries, 0 to 134572
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   booking_id         134573 non-null  object
 1   property_id        134573 non-null  int64
 2   booking_date       134573 non-null  object
 3   check_in_date      134573 non-null  object
 4   checkout_date      134573 non-null  object
 5   no_guests          134573 non-null  float64
 6   room_category      134573 non-null  object
 7   booking_platform   134573 non-null  object
 8   ratings_given      56676 non-null   float64
 9   booking_status     134573 non-null  object
 10  revenue_generated  134573 non-null  int64
 11  revenue_realized   134573 non-null  int64
 12  property_name      134573 non-null  object
 13  category           134573 non-null  object
 14  city               134573 non-null  object
dtypes: float64(2), int64(3), object(10)
memory usage: 16.4+ MB
```

In [80]:
```
df_date.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92 entries, 0 to 91
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      92 non-null     object
 1   mmm yy    92 non-null     object
 2   week no   92 non-null     object
 3   day_type  92 non-null     object
dtypes: object(4)
memory usage: 3.0+ KB
```

In [82]:
```
# Convert the data of the dates from object type to date type:

df_date["date"] = pd.to_datetime(df_date["date"])
```

In [83]:
```
df_date.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92 entries, 0 to 91
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      92 non-null     datetime64[ns]
 1   mmm yy    92 non-null     object
 2   week no   92 non-null     object
 3   day_type  92 non-null     object
dtypes: datetime64[ns](1), object(3)
memory usage: 3.0+ KB
```

In [84]:
```
df_bookings_all["check_in_date"] = pd.to_datetime(df_bookings_all["check_in_date"])
```

In [85]:
```
df_bookings_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 134573 entries, 0 to 134572
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   booking_id         134573 non-null  object
 1   property_id        134573 non-null  int64
 2   booking_date       134573 non-null  object
 3   check_in_date      134573 non-null  datetime64[ns]
 4   checkout_date      134573 non-null  object
 5   no_guests          134573 non-null  float64
 6   room_category      134573 non-null  object
 7   booking_platform   134573 non-null  object
 8   ratings_given      56676 non-null   float64
 9   booking_status     134573 non-null  object
 10  revenue_generated  134573 non-null  int64
 11  revenue_realized   134573 non-null  int64
 12  property_name      134573 non-null  object
 13  category           134573 non-null  object
 14  city               134573 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(3), object(9)
memory usage: 16.4+ MB
```

In [87]:
```
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on="check_in_date",right_on="date")
df_bookings_all
```

Out[87]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_gi
0	May052216558RT11	16558	15-04-22	2022-05-05	7/5/2022	3.0	RT1	tripster	
1	May052216558RT12	16558	30-04-22	2022-05-05	7/5/2022	2.0	RT1	others	N
2	May052216558RT13	16558	1/5/2022	2022-05-05	6/5/2022	3.0	RT1	direct offline	
3	May052216558RT14	16558	3/5/2022	2022-05-05	6/5/2022	2.0	RT1	tripster	
4	May052216558RT15	16558	30-04-22	2022-05-05	10/5/2022	4.0	RT1	others	
...	...	...	...	...	...	...	...	...	
92573	Jul312217564RT45	17564	30-07-22	2022-07-31	1/8/2022	2.0	RT4	others	
92574	Jul312217564RT46	17564	29-07-22	2022-07-31	3/8/2022	1.0	RT4	makeyourtrip	
92575	Jul312217564RT48	17564	30-07-22	2022-07-31	2/8/2022	1.0	RT4	tripster	N
92576	Jul312217564RT49	17564	29-07-22	2022-07-31	1/8/2022	2.0	RT4	logtrip	
92577	Jul312217564RT410	17564	31-07-22	2022-07-31	1/8/2022	2.0	RT4	makeyourtrip	N

92578 rows × 19 columns

In [89]:
```
df_bookings_all.groupby("mmm yy")["revenue_realized"].sum()
```

Out[89]:
```
mmm yy
Jul 22    389940912
Jun 22    377191229
May 22    408375641
Name: revenue_realized, dtype: int64
```

In [93]:
```
# Revenue realized per hotel type

df_bookings_all.category.unique()
```

Out[93]:
```
array(['Luxury', 'Business'], dtype=object)
```

In [99]:
```
df_bookings_all.groupby("category")["revenue_realized"].sum().sort_values(ascending=False)
```

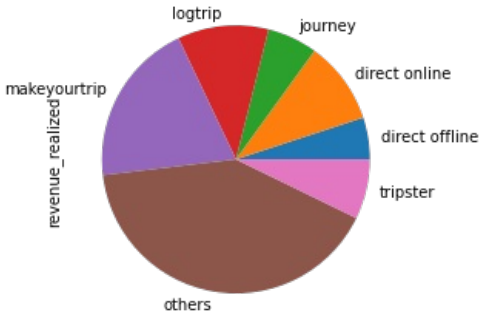
Out[99]:
```
category
Luxury      723557067
Business    451950715
Name: revenue_realized, dtype: int64
```

In [100…
```
# Avg rating per city:

df_bookings_all.groupby("city")["ratings_given"].mean()
```

Out[100]:
```
city
Bangalore    3.403911
Delhi        3.775088
Hyderabad    3.664286
Mumbai       3.644350
Name: ratings_given, dtype: float64
```

In [103…
```
# Pie chart of revenue realized per booking platform

df_bookings_all.groupby("booking_platform")["revenue_realized"].sum().plot(kind="pie")
```

Out[103]:
```
<AxesSubplot:ylabel='revenue_realized'>
```



Overall Insights from the EDA

1. The average occupancy rate in all the room categories is almost the same i.e 58%. There is no significant difference in the average occupancy rate across various room categories.
2. The average occupancy rate in Delhi is the highest i.e 61 % followed by Hyderabad 59 %
3. In the weekends the occupancy rate is 73 % and on weekdays it is 50 %. As it is obvious that people spend holidays in weekends so the occupancy is more in weekends as compared to weekdays.
4. Revenue realized is more from the city Mumbai 668 millions, followed by Bangalore 420 millions then Hyderabad 325 millions and Delhi 294 millions.
5. Revenue realized is more in the month of May 408 millions, followed by July 389 millions and June 377 millions.
6. Revenue realized in Luxury hotel is 723 million which is higher as compared to business hotel 451 millions.
7. Overall average rating for the hotels across various cities is 3.5
8. The platform such as Make your trip, logtrip,and direct online fetches more bookings as compared to other booking sites or options.

Recommendations

1. The average occupancy rate can be increased across room categories and cities by giving special discounts and hassle free booking experience.
2. Average rating should be improved certainly. We recommend average rating to be above 4.2 and this can improved by giving good hospitality, attractive discounts, easy cancellation system, etc. can improve the overall rating.
3. The company should consider attracting more customers for booking from their own website.
4. Efforts should be made to increase occupancy rate even on weekdays. Especially in weekends the occupancy should be 95 % so that more revenue is generated. In weekdays it can be increased to 65 %.

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