AtliQ Hotels Data Analysis Project

==> 1. Data Import and Data Exploration

Datasets

We have 5 csv file

- dim_date.csv
- dim_hotels.csv
- dim_rooms.csv
- fact_aggregated_bookings
- fact_bookings.csv

In [11]:

```
1 import pandas as pd
```

In [46]:

```
1 df_booking=pd.read_csv('Dataset/fact_bookings.csv')
2 df_booking
```

Out[46]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_gu
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	
...	...	...	...	...	...	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	

134590 rows × 12 columns

In [47]:

```
1 df_booking.shape
```

Out[47]:

```
(134590, 12)
```

In [48]:

```
1 df_booking.room_category.unique()
2
```

Out[48]:

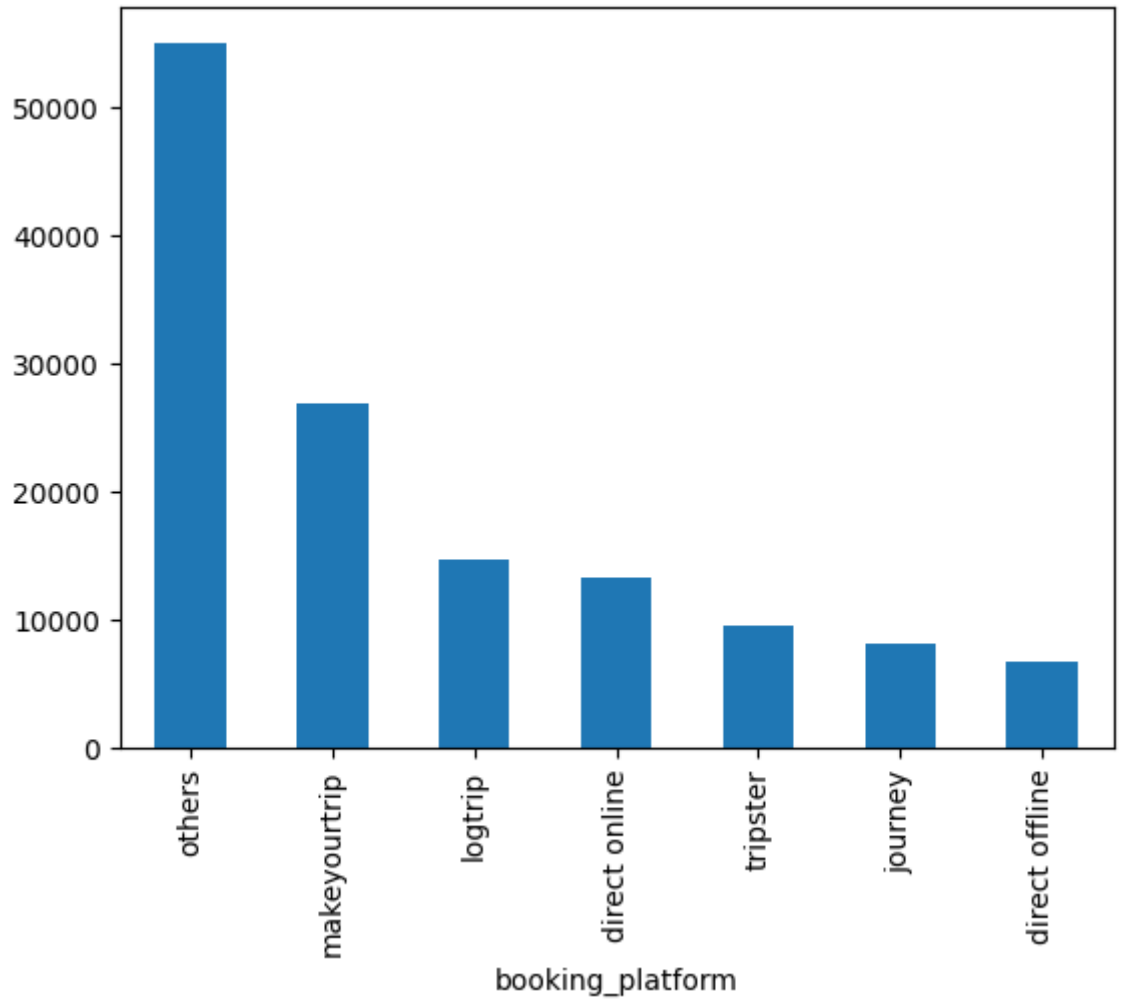
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

In [49]:

```
1 df_booking.booking_platform.value_counts().plot(kind='bar')
```

Out[49]:

```
<Axes: xlabel='booking_platform'>
```



In [50]:

```
df_booking.describe()
```

Out[50]:

	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 [25]:

```
df_date = pd.read_csv('Dataset/dim_date.csv')
df_hotels = pd.read_csv('Dataset/dim_hotels.csv')
df_rooms = pd.read_csv('Dataset/dim_rooms.csv')
df_agg_bookings = pd.read_csv('Dataset/fact_aggregated_bookings.csv')
```

In [27]:

```
df_hotels.shape
```

Out[27]:

```
(25, 4)
```

In [29]:

```
df_hotels.head(3)
```

Out[29]:

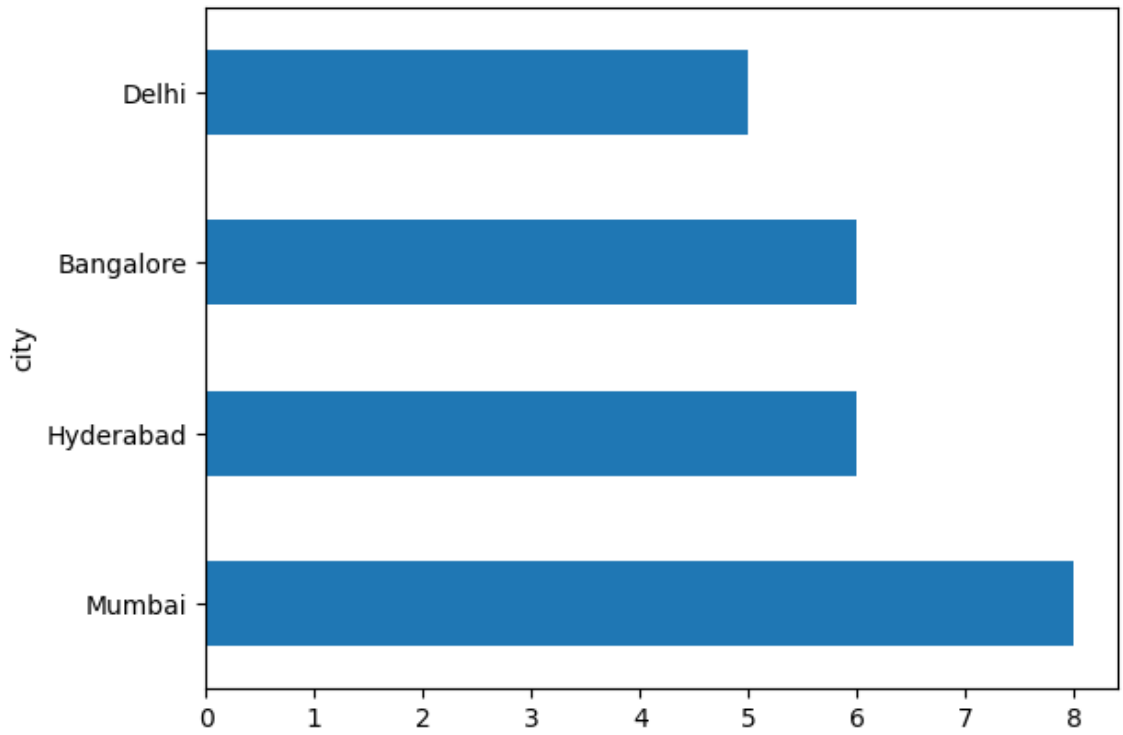
	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi

In [31]:

```
1 df_hotels.city.value_counts().plot(kind='barh')
```

Out[31]:

```
<Axes: ylabel='city'>
```



Exercise-1. Find out unique property ids in aggregate bookings dataset

In [51]:

```
1 df_booking.property_id.unique()
2
```

Out[51]:

```
array([16558, 16559, 16560, 16561, 16562, 16563, 17558, 17559, 17560,
       17561, 17562, 17563, 18558, 18559, 18560, 18561, 18562, 18563,
       19558, 19559, 19560, 19561, 19562, 19563, 17564], dtype=int64)
```

Exercise-2. Find out total bookings per property_id

In [20]:

```
df_agg_book=pd.read_csv('dataset/fact_aggregated_bookings.csv')
df_agg_book
```

Out[20]:

	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
...	...	...	...	...	...
9195	16563	31-Jul-22	RT4	13	18.0
9196	16559	31-Jul-22	RT4	13	18.0
9197	17558	31-Jul-22	RT4	3	6.0
9198	19563	31-Jul-22	RT4	3	6.0
9199	17561	31-Jul-22	RT4	3	4.0

9200 rows × 5 columns

In [31]:

```
df_agg_book.groupby("property_id")['successful_bookings'].sum()
```

Out[31]:

```
property_id
16558    3153
16559    7338
16560    4693
16561    4418
16562    4820
16563    7211
17558    5053
17559    6142
17560    6013
17561    5183
17562    3424
17563    6337
17564    3982
18558    4475
18559    5256
18560    6638
18561    6458
18562    7333
18563    4737
19558    4400
19559    4729
19560    6079
19561    5736
19562    5812
19563    5413
Name: successful_bookings, dtype: int64
```

Exercise-3. Find out days on which bookings are greater than capacity

In [39]:

```
days=df_agg_book[df_agg_book['successful_bookings']>df_agg_book['capacity']]
days

```

Out[39]:

	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

Exercise-4. Find out properties that have highest capacity

In [46]:

```
abc=df_agg_book.capacity.max()
abc

```

Out[46]:

```
50.0
```

In [47]:

```
df_agg_book=df_agg_book[df_agg_book.capacity==df_agg_book.capacity.max()]
df_agg_book
```

Out[47]:

	property_id	check_in_date	room_category	successful_bookings	capacity
27	17558	1-May-22	RT2	38	50.0
128	17558	2-May-22	RT2	27	50.0
229	17558	3-May-22	RT2	26	50.0
328	17558	4-May-22	RT2	27	50.0
428	17558	5-May-22	RT2	29	50.0
...	...	...	...	...	...
8728	17558	27-Jul-22	RT2	22	50.0
8828	17558	28-Jul-22	RT2	21	50.0
8928	17558	29-Jul-22	RT2	23	50.0
9028	17558	30-Jul-22	RT2	32	50.0
9128	17558	31-Jul-22	RT2	30	50.0

92 rows × 5 columns

DATA CLEANING

In [52]:

```
1 df_booking
```

Out[52]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_gu
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	
...	...	...	...	...	...	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	

134590 rows × 12 columns

In [54]:

```
1 df_booking[df_booking.no_guests<0]
```

Out[54]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_g
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	
17924	May122218559RT44	18559	12/5/2022	12/5/2022	14-05-22	
18020	May122218561RT22	18561	8/5/2022	12/5/2022	14-05-22	
18119	May122218562RT311	18562	5/5/2022	12/5/2022	17-05-22	
18121	May122218562RT313	18562	10/5/2022	12/5/2022	17-05-22	
56715	Jun082218562RT12	18562	5/6/2022	8/6/2022	13-06-22	
119765	Jul202219560RT220	19560	19-07-22	20-07-22	22-07-22	
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	

we can see the negative value in number of guest so have to clean the values

In [56]:

```
df_booking=df_booking[df_booking.no_guests>0]
df_booking
```

Out[56]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_gu
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	
...	...	...	...	...	...	
134584	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	

134578 rows × 12 columns

In [57]:

```
df_booking.shape
```

Out[57]:

```
(134578, 12)
```

In [58]:

```
(df_booking.revenue_generated.min(),df_booking.revenue_generated.max())
```

Out[58]:

```
(6500, 28560000)
```

So we can get the information of revenue and find that there is some mistake in max revenue,its too high

In [59]:

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

Out[59]:

```
(15378.036937686695, 93040.1549314641)
```

In [60]:

```
upper_limit=avg+3*std
upper_limit
```

Out[60]:

```
294498.50173207896
```

In [62]:

```
lower_limit=avg-3*std
lower_limit
```

Out[62]:

```
-263742.4278567056
```

In [63]:

```
df_booking[df_booking.revenue_generated>upper_limit]
```

Out[63]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_g
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	
111	May012216559RT32	16559	29-04-22	1/5/2022	2/5/2022	
315	May012216562RT22	16562	28-04-22	1/5/2022	4/5/2022	
562	May012217559RT118	17559	26-04-22	1/5/2022	2/5/2022	
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	

In [65]:

```
df_booking=df_booking[df_booking.revenue_generated<upper_limit]
df_booking
```

Out[65]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_gu
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	
...	...	...	...	...	...	
134584	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	

134573 rows × 12 columns

In [66]:

```
df_booking.revenue_realized.describe()
```

Out[66]:

```
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
```

In [67]:

```
avg,std=(df_booking.revenue_realized.mean(),df_booking.revenue_realized.std())
avg,std
```

Out[67]:

```
(12695.983585117372, 6927.791692242814)
```

In [68]:

```
upper_limit1=avg+3*std
upper_limit1
```

Out[68]:

33479.358661845814

In [69]:

```
lower_limit1=avg-3*std
lower_limit1
```

Out[69]:

-8087.391491611072

In [71]:

```
df_booking[df_booking.revenue_realized>upper_limit1]
```

Out[71]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_g
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	
139	May012216559RT43	16559	1/5/2022	1/5/2022	2/5/2022	
143	May012216559RT47	16559	28-04-22	1/5/2022	3/5/2022	
149	May012216559RT413	16559	24-04-22	1/5/2022	7/5/2022	
222	May012216560RT45	16560	30-04-22	1/5/2022	3/5/2022	
...	...	...	...	...	...	
134328	Jul312219560RT49	19560	31-07-22	31-07-22	2/8/2022	
134331	Jul312219560RT412	19560	31-07-22	31-07-22	1/8/2022	
134467	Jul312219562RT45	19562	28-07-22	31-07-22	1/8/2022	
134474	Jul312219562RT412	19562	25-07-22	31-07-22	6/8/2022	
134581	Jul312217564RT42	17564	31-07-22	31-07-22	1/8/2022	

1299 rows × 12 columns

In [72]:

```
df_rooms
```

Out[72]:

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

In [73]:

```
1 df_booking[df_booking.room_category=='RT4'].revenue_realized.describe()
```

Out[73]:

```
count    16071.000000
mean     23439.308444
std       9048.599076
min       7600.000000
25%      19000.000000
50%      26600.000000
75%      32300.000000
max      45220.000000
Name: revenue_realized, dtype: float64
```

In [76]:

```
1 df_booking.isnull().sum()
```

Out[76]:

```
booking_id               0
property_id              0
booking_date             0
check_in_date            0
checkout_date            0
no_guests                0
room_category            0
booking_platform         0
ratings_given        77897
booking_status           0
revenue_generated        0
revenue_realized         0
dtype: int64
```

Exercise-1. In aggregate bookings find columns that have null values. Fill these null values with whatever you think is the appropriate subtitute (possible ways is to use mean or median)

In [77]:

```
1 df_booking.fillna(0,inplace=True)
```

In [78]:

```
1 df_booking.isnull().sum()
```

Out[78]:

```
booking_id           0
property_id          0
booking_date         0
check_in_date        0
checkout_date        0
no_guests            0
room_category        0
booking_platform     0
ratings_given        0
booking_status       0
revenue_generated    0
revenue_realized     0
dtype: int64
```

Exercise-2. In aggregate bookings find out records that have successful_bookings value greater than capacity. Filter those records

In [80]:

```
1 df_agg_bookings
```

Out[80]:

	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
...	...	...	...	...	...
9195	16563	31-Jul-22	RT4	13	18.0
9196	16559	31-Jul-22	RT4	13	18.0
9197	17558	31-Jul-22	RT4	3	6.0
9198	19563	31-Jul-22	RT4	3	6.0
9199	17561	31-Jul-22	RT4	3	4.0

9200 rows × 5 columns

In [81]:

```
df_agg_bookings[df_agg_bookings.capacity<df_agg_bookings.successful_bookings]
```

Out[81]:

	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

DATA TRANSFORMATION

In [83]:

```
df_agg_bookings.head()
```

Out[83]:

	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 [97]:

```
df_agg_bookings['occupancy_percentage']=df_agg_bookings['successful_bookings']/df_ag
df_agg_bookings['occupancy_percentage']
```

Out[97]:

```
0       0.833333
1       0.933333
2       0.766667
3       1.578947
4       0.947368
          ...
9195    0.722222
9196    0.722222
9197    0.500000
9198    0.500000
9199    0.750000
Name: occupancy_percentage, Length: 9200, dtype: float64
```

In [98]:

```
1 df_agg_bookings
```

Out[98]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occupancy_
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	
...	...	...	...	...	...	
9195	16563	31-Jul-22	RT4	13	18.0	
9196	16559	31-Jul-22	RT4	13	18.0	
9197	17558	31-Jul-22	RT4	3	6.0	
9198	19563	31-Jul-22	RT4	3	6.0	
9199	17561	31-Jul-22	RT4	3	4.0	

9200 rows × 6 columns

In [99]:

```
1 df_agg_bookings['occupancy_percentage']=df_agg_bookings['occupancy_percentage'].appl
2 df_agg_bookings
```

Out[99]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occupancy_
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	
...	...	...	...	...	...	
9195	16563	31-Jul-22	RT4	13	18.0	
9196	16559	31-Jul-22	RT4	13	18.0	
9197	17558	31-Jul-22	RT4	3	6.0	
9198	19563	31-Jul-22	RT4	3	6.0	
9199	17561	31-Jul-22	RT4	3	4.0	

9200 rows × 6 columns

1. What is an average occupancy rate in each of the room categories?

In [100]:

```
1 df_agg_bookings.head()
```

Out[100]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occupancy_per
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 [102]:

```
1 df_agg_bookings.groupby('room_category')['occupancy_percentage'].mean()
```

Out[102]:

```
room_category
RT1    58.224247
RT2    58.040278
RT3    58.028213
RT4    59.300461
Name: occupancy_percentage, dtype: float64
```

I don't understand RT1, RT2 etc. Print room categories such as Standard, Premium, Elite etc along with average occupancy percentage

In [103]:

```
1 df_rooms
```

Out[103]:

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

In [105]:

```
df=pd.merge(df_agg_bookings,df_rooms,left_on="room_category", right_on="room_id")
df.head()
```

Out[105]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occupancy_per
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 [106]:

```
df.drop('room_id',axis=1,inplace=True)
```

In [108]:

```
df.head(4)
```

Out[108]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occupancy_per
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	

2. Print average occupancy rate per city

In [109]:

```
df_hotels.head(4)
```

Out[109]:

	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

In [110]:

```
df=pd.merge(df,df_hotels, on='property_id')
df.head(4)
```

Out[110]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occupancy_per
0	16559	1-May-22	RT1	25	30.0	
1	16559	2-May-22	RT1	20	30.0	
2	16559	3-May-22	RT1	17	30.0	
3	16559	4-May-22	RT1	21	30.0	

In [134]:

```
df.groupby('city')['occupancy_percentage'].mean()
```

Out[134]:

```
city
Bangalore    55.289801
Delhi        60.402877
Hyderabad    56.936423
Mumbai       56.782817
Name: occupancy_percentage, dtype: float64
```

3. When was the occupancy better? Weekday or Weekend?

In [133]:

```
df_date.head(4)
```

Out[133]:

	date	mmm yy	week no	day_type
0	01-May-22	May 22	W 19	weekend
1	02-May-22	May 22	W 19	weekeday
2	03-May-22	May 22	W 19	weekeday
3	04-May-22	May 22	W 19	weekeday

In [135]:

```
1 df.head(4)
```

Out[135]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occupancy_per
0	16559	10-May-22	RT1	18	30.0	
1	16559	10-May-22	RT2	25	41.0	
2	16559	10-May-22	RT3	20	32.0	
3	16559	10-May-22	RT4	13	18.0	

4 rows × 22 columns

In [136]:

```
1 df.groupby('day_type')['occupancy_percentage'].mean()
```

Out[136]:

```
day_type
weekeday    50.903780
weekend     72.393432
Name: occupancy_percentage, dtype: float64
```

Exercise-1. Print revenue realized per hotel type

In [145]:

```
1 df_booking.head(5)
```

Out[145]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0

In [147]:

```
df_hotels.head(4)
```

Out[147]:

	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

In [148]:

```
df_rev=pd.merge(df_booking,df_hotels,on='property_id')
df_rev.head(4)
```

Out[148]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0
3	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0

In [150]:

```
df_rev.groupby('category')['revenue_realized'].mean()
```

Out[150]:

```
category
Business    12880.282693
Luxury      12583.771439
Name: revenue_realized, dtype: float64
```

Exercise-2 Print average rating per city

In [151]:

```
df_rev.groupby('city')['ratings_given'].mean()
```

Out[151]:

```
city
Bangalore    1.427362
Delhi        1.596251
Hyderabad    1.542270
Mumbai       1.540804
Name: ratings_given, dtype: float64
```

Exercise-3 Print a pie chart of revenue realized per booking platform

In [155]:

```
1 df_rev.groupby('booking_platform')['revenue_realized'].sum().plot(kind="pie")
```