India Hotel Data Analyst Project

Datasets in this project*

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

==> 1. Data Import and Data Exploration

Read bookings data in a datagrame

In [140…
```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
```

In [225…
```
df_bookings = pd.read_csv('fact_bookings.csv')
df_bookings.head()
```

Out[225]:

	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

Explore bookings data

In [3]:
```
df_bookings.shape
```

Out[3]:
```
(134590, 12)
```

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

```
Out[4]: Index(['booking_id', 'property_id', 'booking_date', 'check_in_date',
       'checkout_date', 'no_guests', 'room_category', 'booking_platform',
       'ratings_given', 'booking_status', 'revenue_generated',
       'revenue_realized'],
      dtype='object')
```

```
In [5]: df_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134590 entries, 0 to 134589
Data columns (total 12 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   booking_id         134590 non-null  object 
 1   property_id        134590 non-null  int64  
 2   booking_date       134590 non-null  object 
 3   check_in_date      134590 non-null  object 
 4   checkout_date      134590 non-null  object 
 5   no_guests          134587 non-null  float64
 6   room_category      134590 non-null  object 
 7   booking_platform   134590 non-null  object 
 8   ratings_given      56683 non-null   float64
 9   booking_status     134590 non-null  object 
 10  revenue_generated  134590 non-null  int64  
 11  revenue_realized   134590 non-null  int64  
dtypes: float64(2), int64(3), object(7)
memory usage: 12.3+ MB
```

```
In [8]: df_bookings.describe()
```

Out[8]:

	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

Read rest of the files

In [223…
```
df_aggregated_bookings = pd.read_csv('fact_aggregated_bookings.csv')
df_hotels = pd.read_csv('dim_hotels.csv')
df_rooms = pd.read_csv('dim_rooms.csv')
df_date = pd.read_csv('dim_date.csv')
```

Article requirement: Explore aggregate bookings

In [12]:
```
df_aggregated_bookings.head(3)
```

Out[12]:

	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

In [14]:
```
df_aggregated_bookings.shape
```

Out[14]:
```
(9200, 5)
```

Request 1. Find out unique property ids in aggregate bookings dataset

In [15]:
```
df_aggregated_bookings.property_id.unique()
```

Out[15]:
```
array([16559, 19562, 19563, 17558, 16558, 17560, 19558, 19560, 17561,
       16560, 16561, 16562, 16563, 17559, 17562, 17563, 18558, 18559,
       18561, 18562, 18563, 19559, 19561, 17564, 18560])
```

Request-2. Find out total bookings per property_id

In [16]:
```
df_aggregated_bookings.groupby('property_id')['successful_bookings'].sum()
```

```
Out[16]: 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
```

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

```
In [18]: df_aggregated_bookings[df_aggregated_bookings['successful_bookings'] > df_aggregated_bookings['capacity'] ]
```

Out[18]:

	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

Request-4. Find out properties that have highest capacity

```
In [20]: df_aggregated_bookings[df_aggregated_bookings['capacity']== df_aggregated_bookings['capacity'].max()]
```

Out[20]:

	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

==> 2. Data Cleaning

(1) Clean invalid guests

In [23]:
```
df_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134590 entries, 0 to 134589
Data columns (total 12 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   booking_id         134590 non-null  object 
 1   property_id        134590 non-null  int64  
 2   booking_date       134590 non-null  object 
 3   check_in_date      134590 non-null  object 
 4   checkout_date      134590 non-null  object 
 5   no_guests          134587 non-null  float64
 6   room_category      134590 non-null  object 
 7   booking_platform   134590 non-null  object 
 8   ratings_given      56683 non-null   float64
 9   booking_status     134590 non-null  object 
 10  revenue_generated  134590 non-null  int64  
 11  revenue_realized   134590 non-null  int64  
dtypes: float64(2), int64(3), object(7)
memory usage: 12.3+ MB
```

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 [24]:
```
df_bookings[df_bookings.no_guests <= 0]
```

Out[24]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_g
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 [27]:
```
df_fix_guest = df_bookings[df_bookings.no_guests > 0]
df_fix_guest.describe()
```

Out[27]:

	property_id	no_guests	ratings_given	revenue_generated	revenue_realized
count	134578.000000	134578.000000	56679.000000	1.345780e+05	134578.000000
mean	18061.143315	2.036744	3.619048	1.537804e+04	12696.011822
std	1093.053454	1.031710	1.234970	9.304015e+04	6927.841641
min	16558.000000	1.000000	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

(2) Outlier removal in revenue generated

In [33]:
```
# highest_limit = mean + 3*std
df_revenue_avg = df_fix_guest['revenue_generated'].mean()
df_revenue_avg
```

Out[33]:
```
15378.036937686695
```

In [29]:
```
df_fix_guest['revenue_generated'].median()
```

```
Out[29]: 13500.0
```

```
In [32]: df_revenue_std = df_fix_guest['revenue_generated'].std()
         df_revenue_std
```

```
Out[32]: 93040.15493143328
```

```
In [35]: df_highest_limit = df_revenue_avg + 3*df_revenue_std
         df_highest_limit
```

```
Out[35]: 294498.50173198653
```

```
In [36]: #Find out the value in revenue generated that higher than limit standard deviation
         df_fix_guest[df_fix_guest['revenue_generated']>=df_highest_limit]
```

Out[36]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_gi
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	I
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	I
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	2.0	RT2	direct online	

We may see very clearly , there are many revenue generated unreasonable. Hence we need to eliminate this all value and get the real revenue

```
In [39]: df_bookings = df_fix_guest[df_fix_guest['revenue_generated'] < df_highest_limit]
         df_bookings.describe()
```

Out[39]:

	property_id	no_guests	ratings_given	revenue_generated	revenue_realized
count	134573.000000	134573.000000	56676.000000	134573.000000	134573.000000
mean	18061.191658	2.036716	3.619045	14915.959776	12695.983585
std	1093.042273	1.031673	1.234983	6452.676164	6927.791692
min	16558.000000	1.000000	1.000000	6500.000000	2600.000000
25%	17558.000000	1.000000	3.000000	9900.000000	7600.000000
50%	17564.000000	2.000000	4.000000	13500.000000	11700.000000
75%	18563.000000	2.000000	5.000000	18000.000000	15300.000000
max	19563.000000	6.000000	5.000000	45220.000000	45220.000000

Because the value of max revenue generated above is quite higher than the mean and median value (15378.03, 13500) so that we need to find out is this outlier value or not.

In [57]:
```
#Let find out property which have the revenue generated higher than highest limit
highest_limit_revenue_relized = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std()
highest_limit_revenue_relized
```

Out[57]:
```
33479.3586618449
```

In [58]:
```
df_bookings[df_bookings.revenue_realized >= highest_limit_revenue_relized]
```

Out[58]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_g
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	4.0	RT4	others	
139	May012216559RT43	16559	1/5/2022	1/5/2022	2/5/2022	6.0	RT4	tripster	
143	May012216559RT47	16559	28-04-22	1/5/2022	3/5/2022	3.0	RT4	others	
149	May012216559RT413	16559	24-04-22	1/5/2022	7/5/2022	5.0	RT4	logtrip	
222	May012216560RT45	16560	30-04-22	1/5/2022	3/5/2022	5.0	RT4	others	
...	...	...	...	...	...	...	...	...	
134328	Jul312219560RT49	19560	31-07-22	31-07-22	2/8/2022	6.0	RT4	direct online	
134331	Jul312219560RT412	19560	31-07-22	31-07-22	1/8/2022	6.0	RT4	others	
134467	Jul312219562RT45	19562	28-07-22	31-07-22	1/8/2022	6.0	RT4	makeyourtrip	
134474	Jul312219562RT412	19562	25-07-22	31-07-22	6/8/2022	5.0	RT4	direct offline	
134581	Jul312217564RT42	17564	31-07-22	31-07-22	1/8/2022	4.0	RT4	makeyourtrip	

1299 rows × 12 columns

As the table above, we may see the most revenue come from the room which category is RT4. Let's find out. what's kind of room to determine further

In [60]:
```
#Let's chech the table dim_rooms.csv
df_rooms
```

Out[60]:

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

We relized that the room "RT4" is presidential. This kind of room has really high price. But we will determine the maximum revenue will be outlier the price or not. Let's check

In [64]:
```
df_luxury_room = df_bookings[df_bookings['room_category']=='RT4']
df_luxury_room.head(3)
```

Out[64]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given
47	May012216558RT41	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT4	logtrip	NaN
48	May012216558RT42	16558	27-04-22	1/5/2022	2/5/2022	1.0	RT4	tripster	NaN
49	May012216558RT43	16558	29-04-22	1/5/2022	4/5/2022	2.0	RT4	direct offline	NaN

In [ ]:
```
#Calculate the highest limit of this kind of luxury room
```

In [68]:
```
highest_limit_luxury = df_luxury_room['revenue_realized'].mean() + 3*df_luxury_room['revenue_realized'].std()
highest_limit_luxury
```

Out[68]:
```
50585.1056709996
```

In [66]:
```
df_luxury_room.describe()
```

Out[66]:

	property_id	no_guests	ratings_given	revenue_generated	revenue_realized
count	16071.000000	16071.000000	6879.000000	16071.0	16071.000000
mean	18031.070437	2.105283	3.687164	6500.0	23439.308444
std	1034.119639	1.207111	1.266633	0.0	9048.599076
min	16558.000000	1.000000	1.000000	6500.0	7600.000000
25%	17559.000000	1.000000	3.000000	6500.0	19000.000000
50%	18558.000000	2.000000	4.000000	6500.0	26600.000000
75%	18562.000000	2.000000	5.000000	6500.0	32300.000000
max	19563.000000	6.000000	5.000000	6500.0	45220.000000

We may see the maximum of luxury room is 45220 and the highest limit for this is 50585. The maximum is less than highest limit so that we conclude : the maximum price of revenue relized is not outlier.

Working with aggregate booking

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

In [69]:
```
df_aggregated_bookings.head(3)
```

Out[69]:

	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

In [90]:
```
#Check null value
df_aggregated_bookings.isna().sum()
```

Out[90]:
```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               2
dtype: int64
```

In [98]:
```
df_aggregated_bookings_zero_na = df_aggregated_bookings.fillna(0)
```

In [99]:
```
df_aggregated_bookings_zero_na.isna().sum()
```

Out[99]:
```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               0
dtype: int64
```

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

In [100…
```
df1 = df_aggregated_bookings_zero_na
        [df_aggregated_bookings_zero_na['successful_bookings'] > df_aggregated_bookings_zero_na['capacity']]
df1
```

Out[100]:

	property_id	check_in_date	room_category	successful_bookings	capacity
3	17558	1-May-22	RT1	30	19.0
8	17561	1-May-22	RT1	22	0.0
12	16563	1-May-22	RT1	100	41.0
14	17562	1-May-22	RT1	12	0.0
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

==> 3. Data Transformation

Continue working with aggregated booking

Create occupancy percentage column

In [117…
```
df_aggregated_bookings_zero_na['occ_pct'] =
        df_aggregated_bookings_zero_na['successful_bookings']/df_aggregated_bookings_zero_na['capacity']
```

In [121…
```
df_aggregated_bookings_zero_na['occ_pct'] =
        df_aggregated_bookings_zero_na['occ_pct'].apply(lambda x: round(x*100,2))
```

In [123…
```
df_aggregated_bookings_zero_na.head(3)
```

Out[123]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
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

==> 4. Insights Generation

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

In [135…
```
#check the value of occ_pct if any problems cause we make the divide with 0 value
df_aggregated_bookings_zero_na[df_aggregated_bookings_zero_na['capacity']==0]
```

Out[135]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
8	17561	1-May-22	RT1	22	0.0	inf
14	17562	1-May-22	RT1	12	0.0	inf

In [141…

```
# Replace inf to 0
df_aggregated_bookings_zero_na['occ_pct'].replace(np.inf,0, inplace=True)
```

In [142…

```
df_aggregated_bookings_zero_na[df_aggregated_bookings_zero_na['capacity']==0]
```

Out[142]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
8	17561	1-May-22	RT1	22	0.0	0.0
14	17562	1-May-22	RT1	12	0.0	0.0

In [148…

```
df_aggregated_bookings_zero_na.head(3)
```

Out[148]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
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

In [145…

```
df_aggregated_bookings_zero_na.groupby('room_category')['occ_pct'].mean()
```

Out[145]:

```
room_category
RT1    58.173617
RT2    58.040278
RT3    58.028213
RT4    59.300461
Name: occ_pct, dtype: float64
```

Request 2. Print average occupancy rate per city

In [153…

```
#Join with the dim_hotel.csv
df_hotels.head()
```

Out[153]:

	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 [154…
```
df_aggregated_new = pd.merge(df_aggregated_bookings_zero_na, df_hotels, on='property_id')
```

In [155…
```
df_aggregated_new.head()
```

Out[155]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	property_name	category	city
0	16559	1-May-22	RT1	25	30.0	83.33	Atliq Exotica	Luxury	Mumbai
1	16559	1-May-22	RT2	35	41.0	85.37	Atliq Exotica	Luxury	Mumbai
2	16559	1-May-22	RT3	27	32.0	84.38	Atliq Exotica	Luxury	Mumbai
3	16559	1-May-22	RT4	17	18.0	94.44	Atliq Exotica	Luxury	Mumbai
4	16559	2-May-22	RT1	20	30.0	66.67	Atliq Exotica	Luxury	Mumbai

In [157…
```
df_aggregated_new.groupby('city')['occ_pct'].mean().sort_values(ascending=True)
```

Out[157]:
```
city
Bangalore    56.594207
Mumbai       57.896946
Hyderabad    58.144651
Delhi        61.606467
Name: occ_pct, dtype: float64
```

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

In [159…
```
#Let's join with dim_date
df_date.head(5)
```

Out[159]:

	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
4	05-May-22	May 22	W 19	weekeday

In [160…
```
df_date['day_type'].unique()
```

Out[160]:
```
array(['weekend', 'weekeday'], dtype=object)
```

In [161…
```
df_aggregated_new = pd.merge(df_aggregated_new, df_date, left_on='check_in_date', right_on='date')
```

In [162…
```
df_aggregated_new.head()
```

Out[162]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	property_name	category	city	date	mmm yy	wee n
0	16559	10-May-22	RT2	25	41.0	60.98	Atliq Exotica	Luxury	Mumbai	10-May-22	May 22	W 2
1	16559	10-May-22	RT1	18	30.0	60.00	Atliq Exotica	Luxury	Mumbai	10-May-22	May 22	W 2
2	16559	10-May-22	RT3	20	32.0	62.50	Atliq Exotica	Luxury	Mumbai	10-May-22	May 22	W 2
3	16559	10-May-22	RT4	13	18.0	72.22	Atliq Exotica	Luxury	Mumbai	10-May-22	May 22	W 2
4	19562	10-May-22	RT1	18	30.0	60.00	Atliq Bay	Luxury	Bangalore	10-May-22	May 22	W 2

In [164…
```
df_aggregated_new.groupby('day_type')['occ_pct'].mean()
```

Out[164]:
```
day_type
weekeday    50.903780
weekend     72.393432
Name: occ_pct, dtype: float64
```

Request 4: In the month of June, what is the occupancy for different cities

In [165…

```
df_aggregated_new['mmm yy'].unique()
```

Out[165]:

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

In [186…

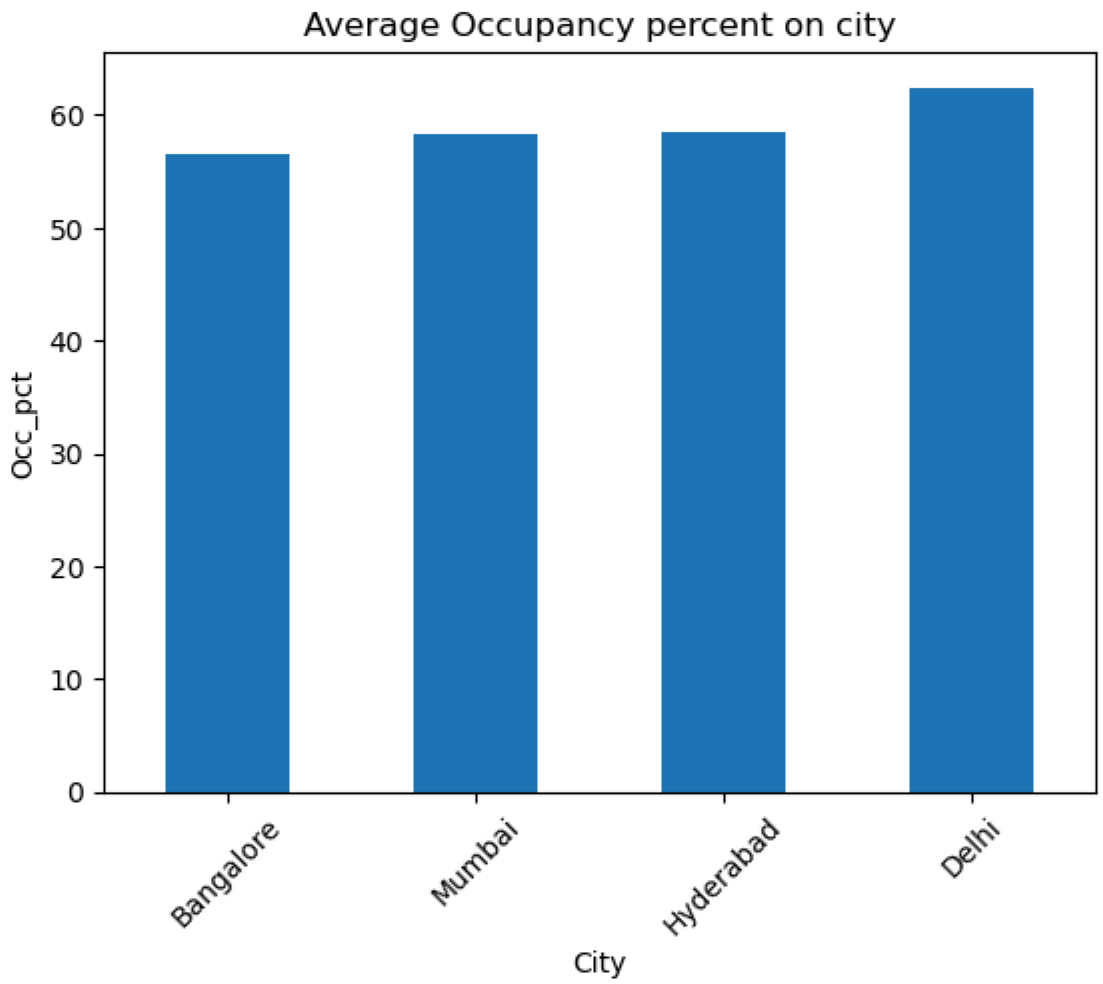
```
mean_june = df_aggregated_new[df_aggregated_new['mmm yy'] == 'Jun 22'].groupby('city')['occ_pct'].mean().
                                                                                    sort_values(ascending=True)

mean_june
```

Out[186]:

```
city
Bangalore    56.578552
Mumbai       58.382560
Hyderabad    58.458075
Delhi        62.474286
Name: occ_pct, dtype: float64
```

In [191…

```
# show the bar chart to compare
mean_june.plot(kind='bar')
plt.xticks(rotation = 45)
plt.title('Average Occupancy percent on city')
plt.xlabel('City')
plt.ylabel('Occ_pct')
plt.show()
```



Request 5: We got new data for the month of august. Append that to existing data

```
In [170…
df_august = pd.read_csv('new_data_august.csv')
df_august.head(4)
```

Out[170]:

	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

In [171…

```
df_august.columns
```

Out[171]:

```
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 [172…

```
df_aggregated_new.columns
```

Out[172]:

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

In [173…

```
#change columns name of new data from august
df_august.rename(columns={
    'occ%' : 'occ_pct',
}, inplace=True)
```

In [174…

```
df_august.head(4)
```

Out[174]:

	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

In [175…

```
df_aggregated_lastest = pd.concat([df_aggregated_new, df_august], ignore_index=True)
```

In [177…

```
df_aggregated_lastest.tail(5)
```

Out[177]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	property_name	category	city	date	mmm yy
6502	19563	01-Aug-22	RT1	23	30.0	76.67	Atliq Palace	Business	Bangalore	NaN	Aug-22
6503	19558	01-Aug-22	RT1	30	40.0	75.00	Atliq Grands	Luxury	Bangalore	NaN	Aug-22
6504	19560	01-Aug-22	RT1	20	26.0	76.92	Atliq City	Business	Bangalore	NaN	Aug-22
6505	17561	01-Aug-22	RT1	18	26.0	69.23	Atliq Blu	Luxury	Mumbai	NaN	Aug-22
6506	17564	01-Aug-22	RT1	10	16.0	62.50	Atliq Seasons	Business	Mumbai	NaN	Aug-22

Working with fact booking data

Request 6. Print revenue realized per city

In [226…

```
df_bookings.head(3)
```

Out[226]:

	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

In [227…

```
#Need to join with dim_hotel to get city
df_hotels.head(3)
```

Out[227]:

	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 [228…

```
df_bookings_hotels = pd.merge(df_bookings, df_hotels, on='property_id')
```

In [229…
```
df_bookings_hotels.head(3)
```

Out[229]:

	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

In [230…
```
df_bookings_hotels.groupby('city')['revenue_realized'].sum().sort_values(ascending = True)
```

Out[230]:
```
city
Delhi        294500318
Hyderabad    325232870
Bangalore    420397050
Mumbai       668640991
Name: revenue_realized, dtype: int64
```

In [231…
```
df_bookings_hotels.shape
```

Out[231]:
```
(134590, 15)
```

Request 7. Print month by month revenue

In [232…
```
#Let's join booking data with dim_date
df_date.head(4)
```

Out[232]:

	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 [233…
```
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 [234…
```
df_bookings_hotels.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134590 entries, 0 to 134589
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   booking_id         134590 non-null  object 
 1   property_id        134590 non-null  int64  
 2   booking_date       134590 non-null  object 
 3   check_in_date      134590 non-null  object 
 4   checkout_date      134590 non-null  object 
 5   no_guests          134587 non-null  float64
 6   room_category      134590 non-null  object 
 7   booking_platform   134590 non-null  object 
 8   ratings_given      56683 non-null   float64
 9   booking_status     134590 non-null  object 
 10  revenue_generated  134590 non-null  int64  
 11  revenue_realized   134590 non-null  int64  
 12  property_name      134590 non-null  object 
 13  category           134590 non-null  object 
 14  city               134590 non-null  object 
dtypes: float64(2), int64(3), object(10)
memory usage: 15.4+ MB
```

We will see the primary key is data and checkout_date. But the type is not the date time. So that we need to exchange to datetime type

In [235…
```
#Change object type from dim_date -> date column to datetime type

df_date['date'] = pd.to_datetime(df_date['date'], dayfirst=True, errors='coerce')
```

```
/var/folders/q3/xgl4pwjd7lbg8skj81tsl0xr0000gn/T/ipykernel_35849/895636221.py:3: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  df_date['date'] = pd.to_datetime(df_date['date'], dayfirst=True, errors='coerce')
```

In [236…
```
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 [243…
```
#Change object type from booking column checkout_date to datetime type

df_bookings_hotels['check_in_date']= pd.to_datetime(df_bookings_hotels['check_in_date'],
                                                                        dayfirst=True, errors = 'coerce')
```

In [244…
```
df_bookings_hotels.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134590 entries, 0 to 134589
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype         
---  ------             --------------   -----         
 0   booking_id         134590 non-null  object        
 1   property_id        134590 non-null  int64         
 2   booking_date       134590 non-null  object        
 3   check_in_date      55804 non-null   datetime64[ns]
 4   checkout_date      134590 non-null  object        
 5   no_guests          134587 non-null  float64       
 6   room_category      134590 non-null  object        
 7   booking_platform   134590 non-null  object        
 8   ratings_given      56683 non-null   float64       
 9   booking_status     134590 non-null  object        
 10  revenue_generated  134590 non-null  int64         
 11  revenue_realized   134590 non-null  int64         
 12  property_name      134590 non-null  object        
 13  category           134590 non-null  object        
 14  city               134590 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(3), object(9)
memory usage: 15.4+ MB
```

In [246…
```
#Join 2 table to get the month
df_bookings = pd.merge(df_bookings_hotels, df_date, left_on='check_in_date', right_on='date')
```

In [247…
```
df_bookings.drop(columns='date', inplace=True)
```

In [250…
```
df_bookings.shape
```

```
Out[250]: (55804, 18)
```

```
In [249…  df_bookings.revenue_realized.count()
```

```
Out[249]: 55804
```

```
In [251…  #Print month by month revenue
          df_bookings.groupby('mmm yy')['revenue_realized'].sum().sort_values(ascending = True)
```

```
Out[251]: mmm yy
          Jun 22    229644140
          May 22    234516453
          Jul 22    243180932
          Name: revenue_realized, dtype: int64
```

Request 8. Print revenue realized per hotel type

```
In [252…  df_bookings.head(3)
```

Out[252]:

	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	2022-05-01	2/5/2022	-3.0	RT1	direct online	1.0
1	May012216558RT12	16558	30-04-22	2022-05-01	2/5/2022	2.0	RT1	others	NaN
2	May012216558RT13	16558	28-04-22	2022-05-01	4/5/2022	2.0	RT1	logtrip	5.0

```
In [253…  df_hotels.head(4)
```

Out[253]:

	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 [254…  df_bookings.groupby('category')['revenue_realized'].sum()
```

```
Out[254]: category
          Business    270682149
          Luxury      436659376
          Name: revenue_realized, dtype: int64
```

Request 9 Print average rating per city

In [256…
```
df_bookings.groupby('city')['ratings_given'].mean().sort_values(ascending = True)
```

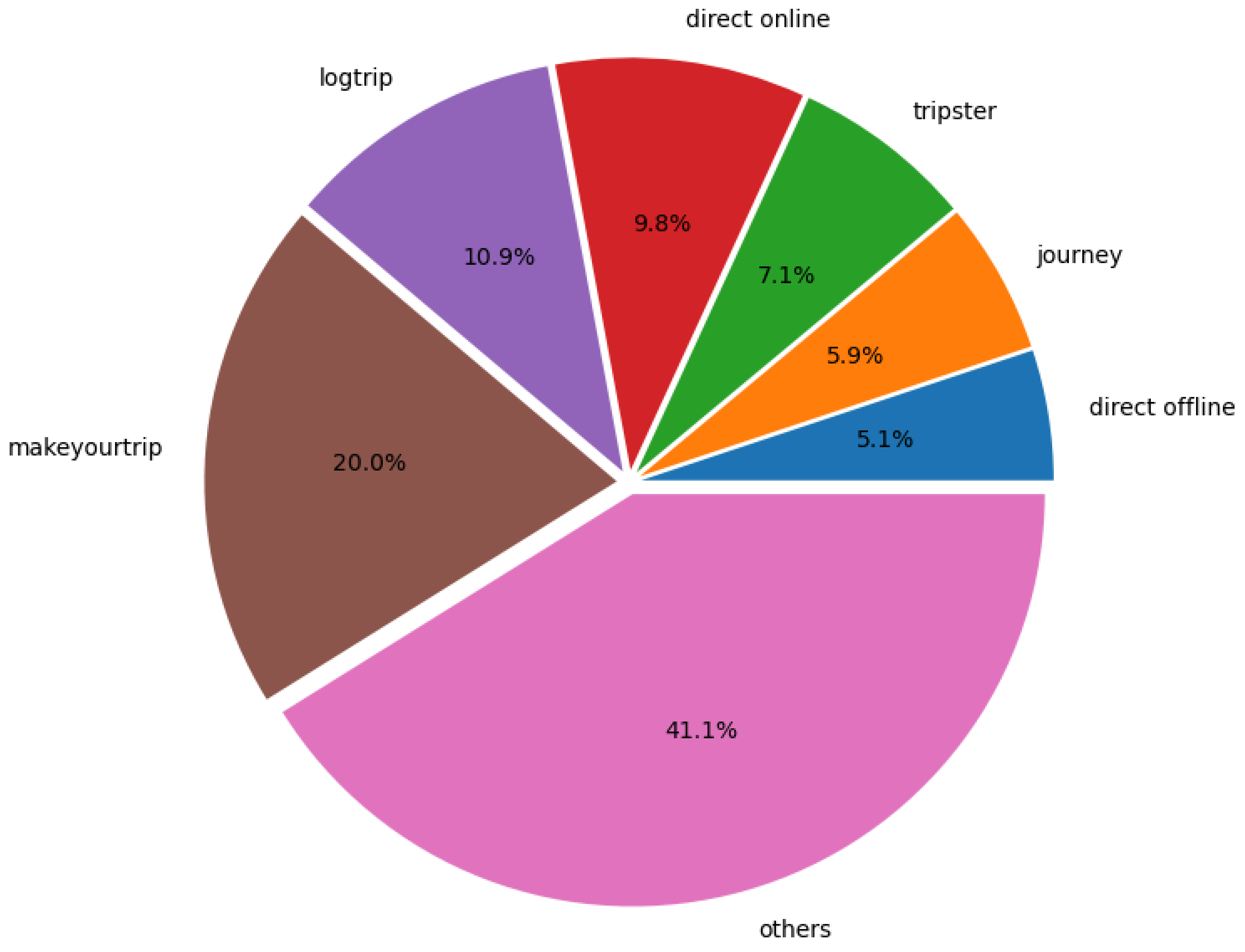
Out[256]:
```
city
Bangalore    3.414599
Hyderabad    3.654123
Mumbai       3.655835
Delhi        3.787587
Name: ratings_given, dtype: float64
```

Request 10 Print a pie chart of revenue realized per booking platform

In [259…
```
df_pie = df_bookings.groupby('booking_platform')['revenue_realized'].sum().sort_values(ascending = True)
df_pie
```

Out[259]:
```
booking_platform
direct offline     36091954
journey            42057593
tripster           50160866
direct online      69257790
logtrip            77084746
makeyourtrip      141736053
others            290952523
Name: revenue_realized, dtype: int64
```

In [272…
```
plt.figure(figsize=(8,8))
plt.pie(df_pie, labels= df_pie.index, autopct='%1.1f%%', explode = (0.03,0.03,0.03,0.03,0.03,0.03,0.03))
plt.show()
```


direct online
logtrip
tripster
9.8%
10.9%
7.1%
journey
5.9%
direct offline
5.1%
makeyourtrip
20.0%
41.1%
others

```
In [ ]:
```

```
In [ ]:
```