AtliQ Hotels Data Analysis Project

> ***AtliQ Hotels***, a luxury hotel chain in India with locations in Mumbai, Delhi, Hyderabad, and Bangalore, is experiencing a decline in business. To address this issue, they have provided a dataset covering three months from May 2022 to July 2022 for analysis, along with separate data for August 2022.
>
> This notebook aims to analyze the data and deliver insights based on the findings.

AGENDA:

- Data Import and Data Exploration
- Data Cleaning
- Data Transformation
- Insights Generation

==> 1. Data Import and Data Exploration

Datasets Available:

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

Importing Necessary Libraries.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Read bookings data in a dataframe

In [2]:
```
df_bookings= pd.read_csv("C:/DataAnalytics/CODEBASICS/Python/Project/source-code/3_p
df_bookings.head(5)
```

Out[2]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cat
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cat
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	

Explore bookings data

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

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

In [4]:
```
df_bookings.room_category.unique()
```

Out[4]:
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

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

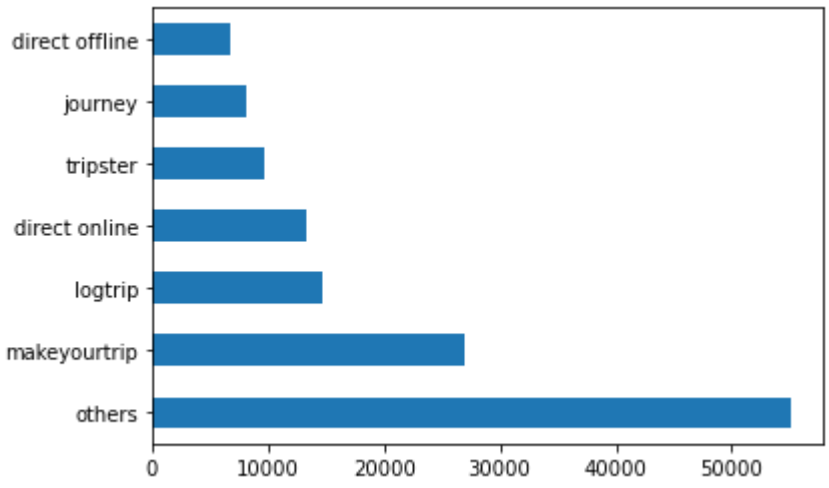
Out[5]:
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

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

Out[6]:
```
others            55066
makeyourtrip      26898
logtrip           14756
direct online     13379
tripster           9630
journey            8106
direct offline     6755
Name: booking_platform, dtype: int64
```

In [7]:
```
df_bookings.booking_platform.value_counts().plot(kind='barh')
```

Out[7]:
```
<AxesSubplot:>
```



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

In [9]:

```
df_date= pd.read_csv("C:/DataAnalytics/CODEBASICS/Python/Project/source-code/3_proje
df_hotels= pd.read_csv("C:/DataAnalytics/CODEBASICS/Python/Project/source-code/3_pro
df_rooms= pd.read_csv("C:/DataAnalytics/CODEBASICS/Python/Project/source-code/3_proj
df_agg_bookings= pd.read_csv("C:/DataAnalytics/CODEBASICS/Python/Project/source-code
```

In [10]:

```
df_date.shape
```

Out[10]:

```
(92, 4)
```

In [11]:

```
df_hotels.shape
```

Out[11]:

```
(25, 4)
```

In [12]:

```
df_hotels.head()
```

Out[12]:

	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 [13]:

```
df_hotels.category.value_counts()
```

Out[13]:

```
Luxury      16
Business     9
Name: category, dtype: int64
```

Exercise: Explore aggregate bookings

In [14]:

```
df_agg_bookings.head(3)
```

Out[14]:

	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

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

In [15]:
```
df_agg_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], dtype=int64)
```

Exercise-2. Find out total bookings per property_id

In [16]:
```
df_agg_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
```

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

In [17]:
```
df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.capacity]
```

Out[17]:

	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

	property_id	check_in_date	room_category	successful_bookings	capacity
9194	18563	31-Jul-22	RT4	20	18.0

Exercise-4. Find out properties that have highest capacity

In [18]:
```
df_agg_bookings.capacity.max()
```

Out[18]:
```
50.0
```

In [19]:
```
df_agg_bookings[df_agg_bookings.capacity==df_agg_bookings.capacity.max()]
```

Out[19]:

	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

In [20]:
```
df_bookings.describe()
```

Out[20]:

	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

(1) Clean invalid guests

In [21]:
```
df_bookings= df_bookings[df_bookings.no_guests>0]
```

In [22]:
```
df_bookings.shape
```

Out[22]:
```
(134578, 12)
```

(2) Outlier removal in revenue generated

In [23]:
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

Out[23]:
```
(6500, 28560000)
```

In [24]:
```
df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median()
```

Out[24]:
```
(15378.036937686695, 13500.0)
```

In [25]:
```
avg, std= df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.std()
```

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]:
```
df_bookings[df_bookings.revenue_generated<0]
```

Out[28]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	b

In [29]:
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

Out[29]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	ro
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	
111	May012216559RT32	16559	29-04-22	1/5/2022	2/5/2022	6.0	
315	May012216562RT22	16562	28-04-22	1/5/2022	4/5/2022	2.0	
562	May012217559RT118	17559	26-04-22	1/5/2022	2/5/2022	2.0	
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	2.0	

In [30]:
```
df_bookings = df_bookings[df_bookings.revenue_generated<=higher_limit]
```

In [31]:
```
df_bookings.shape
```

Out[31]:
```
(134573, 12)
```

In [32]:
```
df_bookings.revenue_realized.describe()
```

Out[32]:
```
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 [33]:
```
higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.
higher_limit
```

Out[33]:
```
33479.3586618449
```

In [34]:
```
df_bookings[df_bookings.revenue_realized>higher_limit]
```

Out[34]:

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

1299 rows × 12 columns

One observation we can have in above dataframe is that all rooms are RT4 which means presidential suit. Now since RT4 is a luxurious room it is likely their rent will be higher. To make a fair analysis, we need to do data analysis only on RT4 room types

In [35]:
```
df_bookings[df_bookings.room_category=="RT4"].revenue_realized.describe()
```

Out[35]:
```
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
```

mean + 3*standard deviation

In [36]:
```
23439+3*9048
```

Out[36]:
```
50583
```

Here higher limit comes to be 50583 and in our dataframe above we can see that max value for revenue realized is 45220.

Hence we can conclude that there is no outlier and we don't need to do any data cleaning on this particular column

In [37]:
```
df_bookings.isnull().sum()
```

Out[37]:
```
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
```

Total values in our dataframe is 134576. Out of that 77899 rows has null rating. Since there are many rows with null rating, we should not filter these values. Also we should not replace this rating with a median or mean rating etc

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 [38]:
```
df_agg_bookings.isnull().sum()
```

Out[38]:
```
property_id           0
check_in_date         0
room_category         0
successful_bookings   0
capacity              2
dtype: int64
```

In [39]:
```
df_agg_bookings [df_agg_bookings.capacity.isna()]
```

Out[39]:

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

In [40]:
```
df_agg_bookings.capacity.median()
```

Out[40]:
```
25.0
```

In [41]:
```
df_agg_bookings.capacity.fillna(df_agg_bookings.capacity.median(), inplace= True)
```

In [42]:
```
df_agg_bookings.loc[[8,15]]
```

Out[42]:

	property_id	check_in_date	room_category	successful_bookings	capacity
8	17561	1-May-22	RT1	22	25.0
15	17563	1-May-22	RT1	21	25.0

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

In [43]:
```
df_agg_bookings[df_agg_bookings.successful_bookings > df_agg_bookings.capacity]
```

Out[43]:

	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

==> 3. Data Transformation

Create occupancy percentage column

In [44]:
```
df_agg_bookings['occ_pct']= df_agg_bookings['successful_bookings']/df_agg_bookings['
df_agg_bookings
```

Out[44]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
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

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
4	16558	1-May-22	RT1	18	19.0	0.947368
...	...	...	...	...	...	...
9195	16563	31-Jul-22	RT4	13	18.0	0.722222
9196	16559	31-Jul-22	RT4	13	18.0	0.722222
9197	17558	31-Jul-22	RT4	3	6.0	0.500000
9198	19563	31-Jul-22	RT4	3	6.0	0.500000
9199	17561	31-Jul-22	RT4	3	4.0	0.750000

9200 rows × 6 columns

Converting the 'occ_pct' into a percentage value

In [45]:
```
df_agg_bookings['occ_pct']= df_agg_bookings['occ_pct'].apply(lambda x: round(x*100,2
```

In [46]:
```
df_agg_bookings.head()
```

Out[46]:

	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
3	17558	1-May-22	RT1	30	19.0	157.89
4	16558	1-May-22	RT1	18	19.0	94.74

==> 4. Insights Generation

---

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

In [47]:
```
df_agg_bookings.groupby('room_category')['occ_pct'].mean()
```

Out[47]:
```
room_category
RT1    58.232748
RT2    58.040278
RT3    58.028213
RT4    59.300461
Name: occ_pct, dtype: float64
```

Instead of RT1, RT2 etc. Print room categories such as Standard, Premium, Elite etc along with average occupancy percentage

In [48]:
```
df= pd.merge(df_agg_bookings, df_rooms, left_on= 'room_category', right_on= 'room_id
```

In [49]:
```
df
```

Out[49]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_id	r
0	16559	1-May-22	RT1	25	30.0	83.33	RT1	
1	19562	1-May-22	RT1	28	30.0	93.33	RT1	
2	19563	1-May-22	RT1	23	30.0	76.67	RT1	
3	17558	1-May-22	RT1	30	19.0	157.89	RT1	
4	16558	1-May-22	RT1	18	19.0	94.74	RT1	
...	...	...	...	...	...	...	...	
9195	16563	31-Jul-22	RT4	13	18.0	72.22	RT4	P
9196	16559	31-Jul-22	RT4	13	18.0	72.22	RT4	P
9197	17558	31-Jul-22	RT4	3	6.0	50.00	RT4	P
9198	19563	31-Jul-22	RT4	3	6.0	50.00	RT4	P
9199	17561	31-Jul-22	RT4	3	4.0	75.00	RT4	P

9200 rows × 8 columns

In [50]:

```
df.drop("room_id",axis=1, inplace=True)
df.head(4)
```

Out[50]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	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

In [51]:

```
df.groupby('room_class')['occ_pct'].mean().round(2)
```

Out[51]:

```
room_class
Elite           58.04
Premium         58.03
Presidential    59.30
Standard        58.23
Name: occ_pct, dtype: float64
```

2. Print average occupancy rate per city

In [52]:

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

Out[52]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	pr
0	16559	1-May-22	RT1	25	30.0	83.33	Standard	
1	16559	2-May-22	RT1	20	30.0	66.67	Standard	
2	16559	3-May-22	RT1	17	30.0	56.67	Standard	
3	16559	4-May-22	RT1	21	30.0	70.00	Standard	

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	pr
4	16559	5-May-22	RT1	16	30.0	53.33	Standard	

In [53]:
```
df.groupby('city')['occ_pct'].mean().round(2)
```

Out[53]:
```
city
Bangalore    56.59
Delhi        61.61
Hyderabad    58.14
Mumbai       57.94
Name: occ_pct, dtype: float64
```

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

In [54]:
```
df_date.head(3)
```

Out[54]:

	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

In [55]:
```
df= pd.merge(df,df_date, left_on= 'check_in_date', right_on= 'date')
```

In [56]:
```
df.head()
```

Out[56]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	pr
0	16559	10-May-22	RT1	18	30.0	60.00	Standard	
1	16559	10-May-22	RT2	25	41.0	60.98	Elite	
2	16559	10-May-22	RT3	20	32.0	62.50	Premium	
3	16559	10-May-22	RT4	13	18.0	72.22	Presidential	
4	19562	10-May-22	RT1	18	30.0	60.00	Standard	

In [57]:
```
df.groupby('day_type')['occ_pct'].mean().round(2)
```

Out[57]:
```
day_type
weekeday    50.90
```

```
weekend     72.39
Name: occ_pct, dtype: float64
```

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

In [58]:
```
df_june22= df[df['mmm yy']=='Jun 22']
df_june22.head(3)
```

Out[58]:

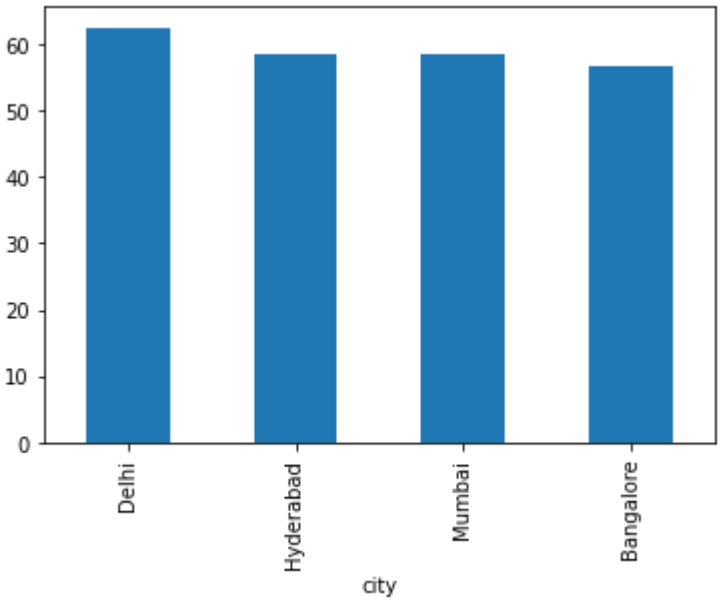
	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard
2201	16559	10-Jun-22	RT2	26	41.0	63.41	Elite
2202	16559	10-Jun-22	RT3	20	32.0	62.50	Premium

In [59]:
```
df_june22.groupby('city')['occ_pct'].mean().round(2).sort_values(ascending= False)
```

Out[59]:
```
city
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Bangalore    56.58
Name: occ_pct, dtype: float64
```

In [60]:
```
df_june22.groupby('city')['occ_pct'].mean().round(2).sort_values(ascending= False).p
```

Out[60]:
```
<AxesSubplot:xlabel='city'>
```



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

In [61]:
```
df_august= pd.read_csv('C:/DataAnalytics/CODEBASICS/Python/Project/source-code/3_pro
df_august.head(3)
```

Out[61]:

	property_id	property_name	category	city	room_category	room_class	check_in_date	mmm y
0	16559	Atliq Exotica	Luxury	Mumbai	RT1	Standard	01-Aug-22	Aug 2
1	19562	Atliq Bay	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug 2
2	19563	Atliq Palace	Business	Bangalore	RT1	Standard	01-Aug-22	Aug 2

In [62]:

```
latest_df= pd.concat([df,df_august], ignore_index=True, axis=0)
latest_df.tail(10)
```

Out[62]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class
6497	18560	31-Jul-22	RT2	34	40.0	85.00	Elite
6498	18560	31-Jul-22	RT3	17	24.0	70.83	Premium
6499	18560	31-Jul-22	RT4	12	15.0	80.00	Presidential
6500	16559	01-Aug-22	RT1	30	30.0	NaN	Standard
6501	19562	01-Aug-22	RT1	21	30.0	NaN	Standard
6502	19563	01-Aug-22	RT1	23	30.0	NaN	Standard
6503	19558	01-Aug-22	RT1	30	40.0	NaN	Standard
6504	19560	01-Aug-22	RT1	20	26.0	NaN	Standard
6505	17561	01-Aug-22	RT1	18	26.0	NaN	Standard
6506	17564	01-Aug-22	RT1	10	16.0	NaN	Standard

6. Print revenue realized per city

In [63]:

```
df_bookings.head()
```

Out[63]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cat
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	

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cat
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 [64]:
```
df_hotels.head(3)
```

Out[64]:

	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 [65]:
```
df_bookings_all = pd.merge(df_bookings, df_hotels, on="property_id")
df_bookings_all.head(3)
```

Out[65]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cat
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	

In [66]:
```
df_bookings_all.groupby("city")["revenue_realized"].sum()
```

Out[66]:
```
city
Bangalore    420383550
Delhi        294404488
Hyderabad    325179310
Mumbai       668569251
Name: revenue_realized, dtype: int64
```

7. Print month by month revenue

In [67]:
```
df_date.head(3)
```

Out[67]:

	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

In [68]:
```
df_bookings_all.head(3)
```

Out[68]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cat
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cat
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	

In [69]:
```
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 [70]:
```
df_date["date"] = pd.to_datetime(df_date["date"])
df_date.head(3)
```

Out[70]:

	date	mmm yy	week no	day_type
0	2022-05-01	May 22	W 19	weekend
1	2022-05-02	May 22	W 19	weekeday
2	2022-05-03	May 22	W 19	weekeday

In [71]:
```
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 [72]:
```
df_bookings_all["check_in_date"] = pd.to_datetime(df_bookings_all["check_in_date"])
df_bookings_all.head(4)
```

Out[72]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cat
0	May012216558RT12	16558	30-04-22	2022-01-05	2/5/2022	2.0	
1	May012216558RT15	16558	27-04-22	2022-01-05	2/5/2022	4.0	
2	May012216558RT16	16558	1/5/2022	2022-01-05	3/5/2022	2.0	
3	May012216558RT17	16558	28-04-22	2022-01-05	6/5/2022	2.0	

In [73]:
```
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on="check_in_date", right_
df_bookings_all.head(3)
```

Out[73]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cat
0	May052216558RT11	16558	15-04-22	2022-05-05	7/5/2022	3.0	
1	May052216558RT12	16558	30-04-22	2022-05-05	7/5/2022	2.0	
2	May052216558RT13	16558	1/5/2022	2022-05-05	6/5/2022	3.0	

In [74]:
```
df_bookings_all.groupby("mmm yy")["revenue_realized"].sum()
```

Out[74]:
```
mmm yy
Jul 22    389940912
Jun 22    377191229
May 22    408375641
Name: revenue_realized, dtype: int64
```

Exercise-1. Print revenue realized per hotel type

In [75]:
```
df_bookings_all.property_name.unique()
```

Out[75]:
```
array(['Atliq Grands', 'Atliq Exotica', 'Atliq City', 'Atliq Blu',
       'Atliq Bay', 'Atliq Palace', 'Atliq Seasons'], dtype=object)
```

In [76]:
```
df_bookings_all.groupby('property_name')['revenue_realized'].sum(),round(2)
```

Out[76]:
```
(property_name
 Atliq Bay        179416721
 Atliq Blu        179203544
 Atliq City       196555383
 Atliq Exotica    219076161
 Atliq Grands     145860641
 Atliq Palace     209474575
 Atliq Seasons     45920757
 Name: revenue_realized, dtype: int64,
 2)
```

In [77]:
```
df_bookings_all.groupby("property_name")["revenue_realized"].sum().round(2).sort_val
```

Out[77]:
```
property_name
Atliq Seasons     45920757
Atliq Grands     145860641
```

```
Atliq Blu        179203544
Atliq Bay        179416721
Atliq City       196555383
Atliq Palace     209474575
Atliq Exotica    219076161
Name: revenue_realized, dtype: int64
```

Exercise-2 Print average rating per city

In [78]:
```
df_bookings_all.groupby('city')['ratings_given'].mean().round(2)
```

Out[78]:
```
city
Bangalore    3.40
Delhi        3.78
Hyderabad    3.66
Mumbai       3.64
Name: ratings_given, dtype: float64
```

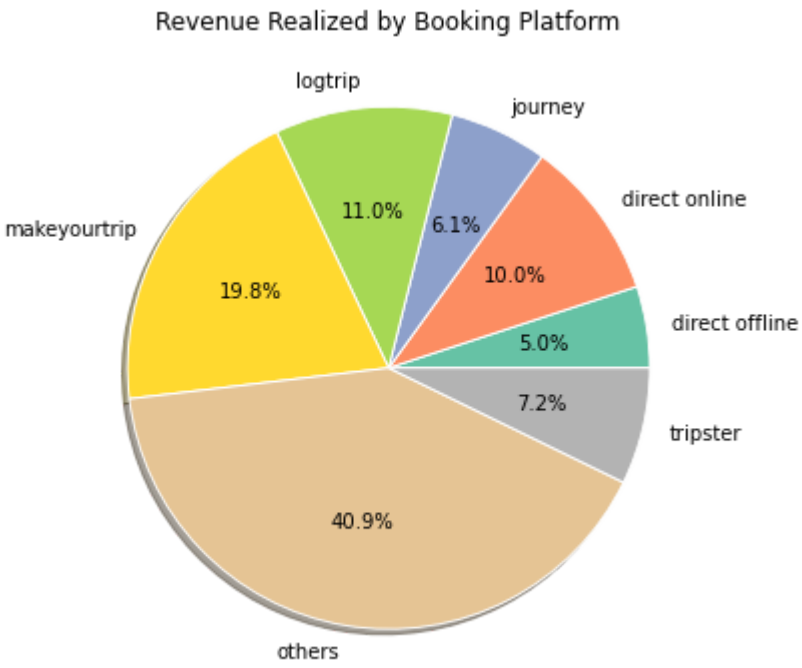
Exercise-3 Print a pie chart of revenue realized per 'booking' platform

In [196…
```
df_bookings_all.groupby('booking_platform')['revenue_realized'].sum().sort_values(as
```

Out[196…
```
booking_platform
others            480698244
makeyourtrip      233132708
logtrip           129036321
direct online     117245053
tripster           84865013
journey            71231599
direct offline     59298844
Name: revenue_realized, dtype: int64
```

In [95]:
```
data = df_bookings_all.groupby('booking_platform')['revenue_realized'].sum()

# Plot
data.plot.pie(
    autopct='%1.1f%%',
    figsize=(10, 6),
    colormap='Set2',
    ylabel='',
    shadow= True,
    wedgeprops={'edgecolor': 'white'},
    title='Revenue Realized by Booking Platform'
)
plt.show()
```


Revenue Realized by Booking Platform
logtrip
journey
direct online
direct offline
tripster
others
makeyourtrip
11.0%
6.1%
10.0%
5.0%
7.2%
40.9%
19.8%

In [ ]:

In [ ]: