AtliQ Hotels Data Analysis Project

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

Data Analytics Project Steps

1. Understanding the business Problem
2. Data Collection and Understanding
3. Data cleaning and Exploration
4. Data Transformation
5. Collect Insights

==> 1. Data Import and Extraction

Datasets

We have 5 CSV files

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

Read bookings data in a dataframe

In [2]:
```
df_bookings = pd.read_csv(r"C:\Users\HP\Projects - Python\fact_bookings.csv")
```

In [3]:
```
df_bookings.head(5)
```

Out[3]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
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)
```

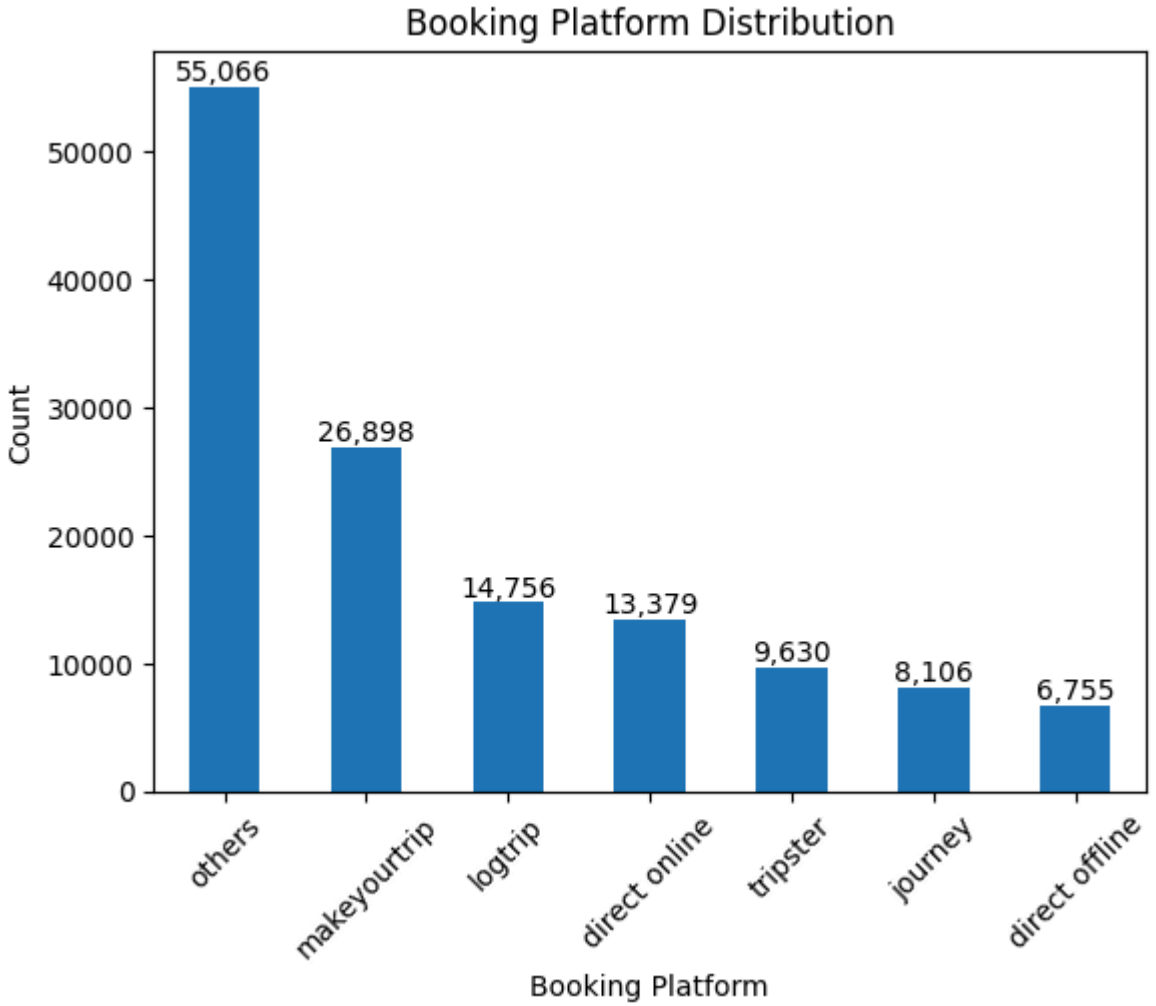
In [13]:
```
room_category_count = df_bookings.room_category.value_counts()
room_category_count
```

Out[13]:
```
room_category
RT2    49505
RT1    38446
RT3    30566
RT4    16073
Name: count, dtype: int64
```

In [14]:
```
booking_platform_count = df_bookings.booking_platform.value_counts()
booking_platform_count
```

Out[14]:
```
booking_platform
others            55066
makeyourtrip      26898
logtrip           14756
direct online     13379
tripster           9630
journey            8106
direct offline     6755
Name: count, dtype: int64
```

In [15]:
```
import matplotlib.pyplot as plt

ax = df_bookings.booking_platform.value_counts().sort_values(ascending=False).plot(kind="bar")

#Add data Labels
for p in ax.patches:
    ax.annotate(f"{p.get_height():,.0f}",
                (p.get_x() + p.get_width()/2, p.get_height()),
                ha = 'center', va = 'bottom', fontsize = 10, color = 'black')

plt.xticks(rotation= 45)
plt.xlabel("Booking Platform")
plt.ylabel("Count")
plt.title("Booking Platform Distribution")

plt.show()
```



Booking Statistics

In [16]:
```
df_bookings.describe()
```

Out[16]:

	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 [18]:
```
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
```

Read Hotels Data in Dataframe

In [21]:
```
df_hotels = pd.read_csv(r"C:\Users\HP\Projects - Python\dim_hotels.csv")
df_hotels.head()
```

Out[21]:

	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 [24]:
```
df_hotels.shape
```

Out[24]:
```
(25, 4)
```

In [25]:
```
df_hotels.category.unique()
```

Out[25]:
```
array(['Luxury', 'Business'], dtype=object)
```

In [26]:
```
df_hotels.city.unique()
```

Out[26]:
```
array(['Delhi', 'Mumbai', 'Hyderabad', 'Bangalore'], dtype=object)
```

In [27]:
```
df_hotels.category.value_counts()
```

Out[27]:
```
category
Luxury      16
Business     9
Name: count, dtype: int64
```

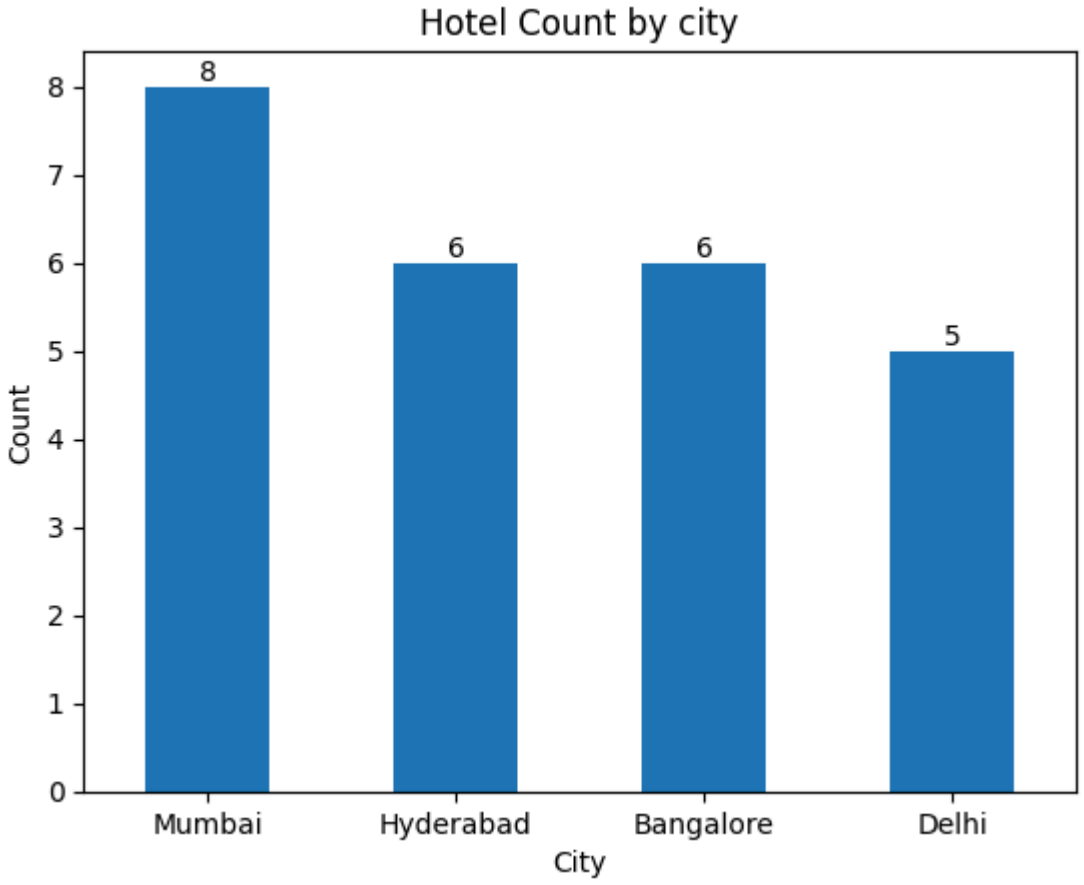
In [28]:
```
df_hotels.city.value_counts()
```

Out[28]:
```
city
Mumbai       8
```

```
Hyderabad    6
Bangalore    6
Delhi        5
Name: count, dtype: int64
```

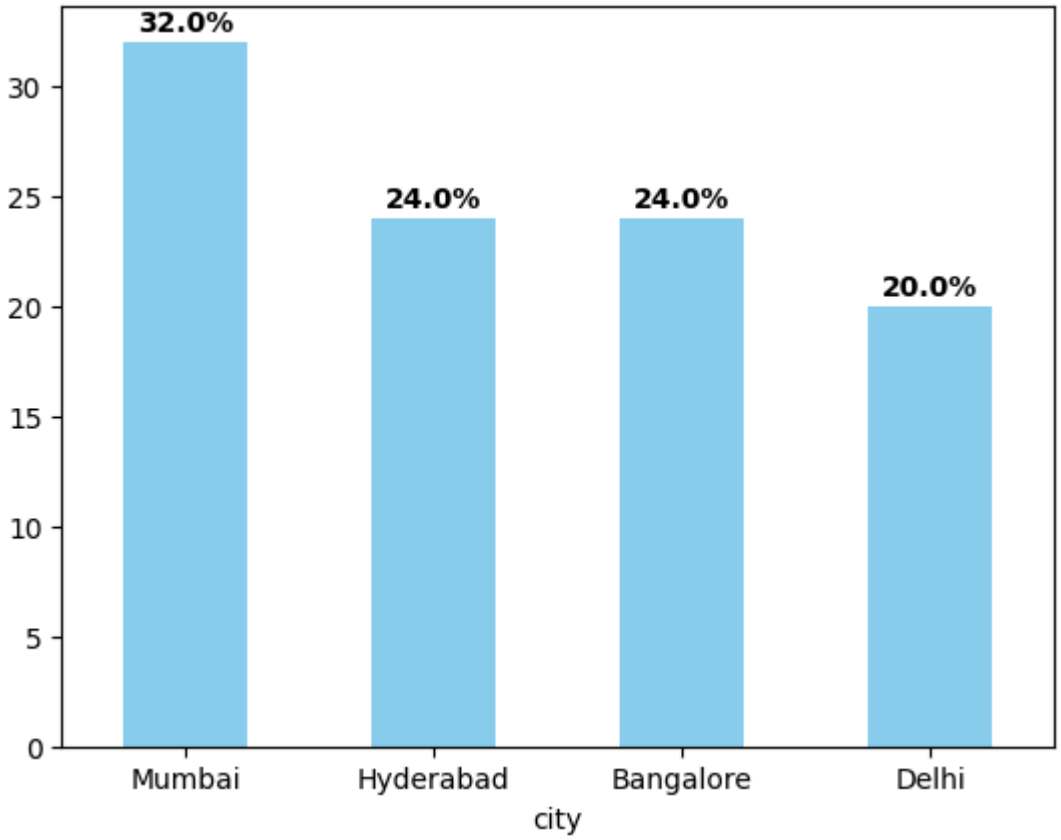
In [29]:

```python
import matplotlib.pyplot as plt

#create the bar plot
ax = df_hotels.city.value_counts().plot(kind= "bar")

# Add data labels
for p in ax.patches:
    ax.annotate(f"{p.get_height():,}",
                (p.get_x() + p.get_width()/2, p.get_height()),
                ha = "center", va = 'bottom', fontsize= 10, color='black')


#Formatting
plt.xticks(rotation = 0) # keep x-axis labels horizontal
plt.xlabel("City")
plt.ylabel("Count")
plt.title("Hotel Count by city")

plt.show()
```



In [31]:
```
# Now convert labels as percentages
import matplotlib.pyplot as plt

#Calculate value counts and percentages
city_counts = df_hotels.city.value_counts()
total = city_counts.sum()
percentages = (city_counts/total) * 100 #convert to percentage

# Plot the bar chart
ax = percentages.plot(kind ="bar", color = "skyblue")

# Add percentages label to each bar
for i, v in enumerate(percentages):
```

```
    plt.text(i, v + 0.5, f"{v:.1f}%", ha = "center", fontsize = 10, fontweight = 'bold')

# Rotate x-axis labels
plt.xticks(rotation = 0)

# show plot
plt.show()
```



Read aggregated_bookings data in dataframe

In [33]:
```
fact_agg_bookings = pd.read_csv(r"C:\Users\HP\Projects - Python\fact_aggregated_bookings.csv")
fact_agg_bookings.head(5)
```

Out[33]:

	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 [34]:
```
type(fact_agg_bookings)
```

Out[34]:
```
pandas.core.frame.DataFrame
```

In [35]:
```
# 1. Find out unique property ids in aggregated bookings datasets

fact_agg_bookings.property_id.unique()
```

Out[35]:
```
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)
```

In [36]:
```
# 2. Find out total bookings per property_id
fact_agg_bookings.groupby("property_id")["successful_bookings"].sum()
```

Out[36]:
```
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
```

In [37]:
```
# 3. Find out days on which bookings are greater than capacity

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

Out[37]:

	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 [38]:
```
df_bookings.describe()
```

Out[38]:

	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 [39]:
```
df_bookings[df_bookings.no_guests<0]
```

Out[39]:

	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	I
18020	May122218561RT22	18561	8/5/2022	12/5/2022	14-05-22	-12.0	RT2	makeyourtrip	I
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	I
56715	Jun082218562RT12	18562	5/6/2022	8/6/2022	13-06-22	-17.0	RT1	others	I
119765	Jul202219560RT220	19560	19-07-22	20-07-22	22-07-22	-1.0	RT2	others	I
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	-4.0	RT4	logtrip	

In [40]:
```
df_bookings.shape
```

Out[40]: (134590, 12)

In [41]:
```
df_bookings = df_bookings[df_bookings.no_guests>0]
df_bookings
```

Out[41]:

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

134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	1.0	RT4	tripster	N
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	N

134578 rows × 12 columns

In [42]:
```
df_bookings.shape
```

Out[42]: (134578, 12)

In [43]:
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

Out[43]: (6500, 28560000)

In [44]:
```
avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.std()
avg, std
```

Out[44]: (15378.036937686695, 93040.1549314641)

In [45]:
```
higher_limit = avg + 3*std
higher_limit
```

Out[45]: 294498.50173207896

In [46]:
```
lower_limit = avg - 3*std
lower_limit
```

Out[46]: -263742.4278567056

In [47]:
```
df_bookings[df_bookings.revenue_generated<0]
```

Out[47]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat

In [58]:
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

Out[58]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_g
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
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	4.0	RT4	others	I
139	May012216559RT43	16559	1/5/2022	1/5/2022	2/5/2022	6.0	RT4	tripster	
149	May012216559RT413	16559	24-04-22	1/5/2022	7/5/2022	5.0	RT4	logtrip	I
...	...	...	...	...	...	...	...	...	
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	
134469	Jul312219562RT47	19562	10/7/2022	31-07-22	6/8/2022	5.0	RT4	makeyourtrip	I
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	

901 rows × 12 columns

In [49]:
```
df_bookings[df_bookings.revenue_generated < higher_limit]
df_bookings
```

Out[49]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_giv
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	N
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	N
...	...	...	...	...	...	...	...	...	
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	N
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	N

134578 rows × 12 columns

In [54]:
```
df_bookings.shape
```

Out[54]:
```
(134578, 12)
```

In [59]:
```
df_bookings.revenue_realized.describe()
```

Out[59]:
```
count    134578.000000
mean      12696.011822
std        6927.841641
min        2600.000000
25%        7600.000000
50%       11700.000000
75%       15300.000000
max       45220.000000
Name: revenue_realized, dtype: float64
```

In [60]:
```
higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std()
higher_limit
```

Out[60]:
```
33479.53674501214
```

In [61]:
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

Out[61]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_giv
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	N
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	4.0	RT4	others	N
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	
...	...	...	...	...	...	...	...	...	
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	
134469	Jul312219562RT47	19562	10/7/2022	31-07-22	6/8/2022	5.0	RT4	makeyourtrip	N
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	

1733 rows × 12 columns

In [63]:
```
df_rooms = pd.read_csv(r"C:\Users\HP\Projects - Python\dim_rooms.csv")
df_rooms
```

Out[63]:

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

In [64]:
```
df_bookings[df_bookings.room_category=="RT4"].revenue_realized.describe()
```

Out[64]:
```
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 [ ]:
```

```

In [65]:
```
df_bookings.isnull().sum()
```

Out[65]:
```
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        77899
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**

In [66]:
```
# In aggregate bookings find columns that have null values. Fill these null values with whatever you think is the
# appropriate substitute (possible ways is to use mean or median)

fact_agg_bookings.isnull().sum()
```

Out[66]:
```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               2
dtype: int64
```

In [68]:
```
fact_agg_bookings[fact_agg_bookings.capacity.isna()]
```

Out[68]:

	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 [69]:
```
fact_agg_bookings.capacity.median()
```

Out[69]:
```
25.0
```

In [70]:
```
fact_agg_bookings.capacity.fillna(fact_agg_bookings.capacity.median(), inplace=True)
```

In [71]:
```
fact_agg_bookings.loc[[8,15]]
```

Out[71]:

	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

In [72]:
```
# In aggregate bookings find out records that have successful_bookings value greater than capacity. Filter those rec
fact_agg_bookings[fact_agg_bookings.successful_bookings > fact_agg_bookings.capacity]
```

Out[72]:

	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 [73]:
```
fact_agg_bookings.shape
```

Out[73]: (9200, 5)

In [77]:
```
fact_agg_bookings = fact_agg_bookings[fact_agg_bookings.successful_bookings<=fact_agg_bookings.capacity]
fact_agg_bookings.shape
```

Out[77]: (9194, 5)

==> 3. Data Transformation

In [83]:
```
fact_agg_bookings.head()
```

Out[83]:

	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	16558	1-May-22	RT1	18	19.0	94.74
5	17560	1-May-22	RT1	28	40.0	70.00

In [84]:
```
fact_agg_bookings["occ_pct"] = fact_agg_bookings["successful_bookings"]/fact_agg_bookings["capacity"]
fact_agg_bookings.head()
```

Out[84]:

	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
4	16558	1-May-22	RT1	18	19.0	0.947368
5	17560	1-May-22	RT1	28	40.0	0.700000

In [85]:
```
fact_agg_bookings["occ_pct"] = fact_agg_bookings["occ_pct"].apply(lambda x:round(x*100, 2))
fact_agg_bookings.head()
```

Out[85]:

	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	16558	1-May-22	RT1	18	19.0	94.74
5	17560	1-May-22	RT1	28	40.0	70.00

In [86]:
```
df_bookings.head()
```

Out[86]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
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
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN

In [87]:
```
fact_agg_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9194 entries, 0 to 9199
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   property_id          9194 non-null   int64  
 1   check_in_date        9194 non-null   object 
 2   room_category        9194 non-null   object 
 3   successful_bookings  9194 non-null   int64  
 4   capacity             9194 non-null   float64
 5   occ_pct              9194 non-null   float64
dtypes: float64(2), int64(2), object(2)
memory usage: 502.8+ KB
```

There are various types of data transformations that you may have to perform based on the need. Few examples of data transformations are,

1. Creating new columns
2. Normalization
3. Merging data
4. Aggregation

==> 4. Insights Generation

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

In [88]:
```
fact_agg_bookings.groupby("room_category")["occ_pct"].mean()
```

Out[88]:
```
room_category
RT1    57.889643
RT2    58.009756
RT3    58.028213
RT4    59.277925
Name: occ_pct, dtype: float64
```

In [89]:
```
df_rooms
```

Out[89]:

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

In [90]:
```
df = pd.merge(fact_agg_bookings, df_rooms, left_on="room_category",right_on= "room_id")
df.head()
```

Out[90]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	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	16558	1-May-22	RT1	18	19.0	94.74	RT1	Standard
4	17560	1-May-22	RT1	28	40.0	70.00	RT1	Standard

In [93]:
```
df.groupby("room_class")["occ_pct"].mean().round(2)
```

Out[93]:
```
room_class
Elite           58.01
Premium         58.03
Presidential    59.28
Standard        57.89
Name: occ_pct, dtype: float64
```

In [94]:
```
df.drop("room_id", axis = 1, inplace= True)
df.head(5)
```

Out[94]:

	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	16558	1-May-22	RT1	18	19.0	94.74	Standard
4	17560	1-May-22	RT1	28	40.0	70.00	Standard

2. Print average occupancy rate per city

In [95]:
```
df_hotels.head(4)
```

Out[95]:

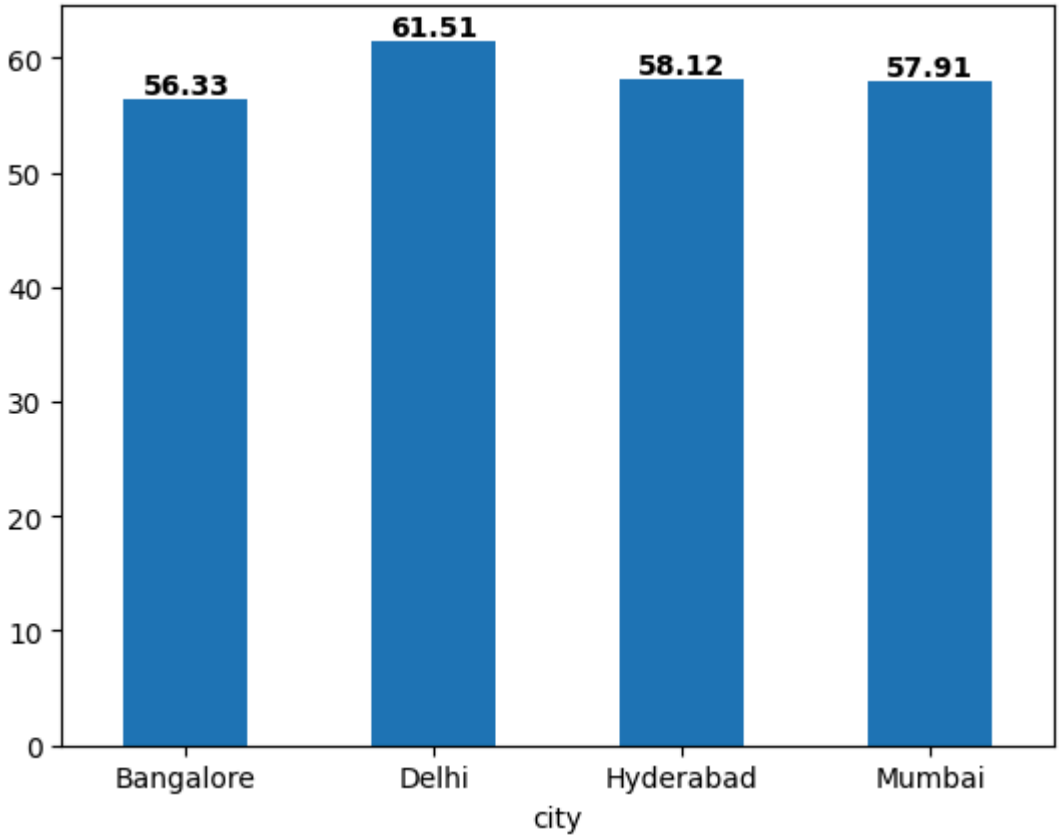
	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi
3	16561	Atliq Blu	Luxury	Delhi

In [97]:
```
df = pd.merge(df, df_hotels, on = "property_id")
df.head()
```

Out[97]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city
0	16559	1-May-22	RT1	25	30.0	83.33	Standard	Atliq Exotica	Luxury	Mumbai
1	16559	2-May-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury	Mumbai
2	16559	3-May-22	RT1	17	30.0	56.67	Standard	Atliq Exotica	Luxury	Mumbai
3	16559	4-May-22	RT1	21	30.0	70.00	Standard	Atliq Exotica	Luxury	Mumbai
4	16559	5-May-22	RT1	16	30.0	53.33	Standard	Atliq Exotica	Luxury	Mumbai

In [98]:
```
import matplotlib .pyplot as plt

# Grouping and plotting
ax = df.groupby("city")["occ_pct"].mean().round(2).plot(kind="bar")

# Adding data Labels
for p in ax.patches:
    ax.annotate(f'{p.get_height()}',
                (p.get_x() + p.get_width() / 2, p.get_height()),
                ha = 'center', va = 'bottom', fontsize = 10, fontweight = 'bold')

# Rotating x-axis labels
plt.xticks(rotation = 0)

#show plot
plt.show()
```



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

In [101…

```
df_date = pd.read_csv(r"C:\Users\HP\Projects - Python\dim_date.csv")
df_date.head()
```

Out[101…

	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 [103…
```
df = pd.merge(df, df_date, left_on= "check_in_date", right_on="date")
df.head(3)
```

Out[103…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	date	m
0	16559	10-May-22	RT1	18	30.0	60.00	Standard	Atliq Exotica	Luxury	Mumbai	10-May-22	
1	16559	10-May-22	RT2	25	41.0	60.98	Elite	Atliq Exotica	Luxury	Mumbai	10-May-22	
2	16559	10-May-22	RT3	20	32.0	62.50	Premium	Atliq Exotica	Luxury	Mumbai	10-May-22	

In [104…
```
df.groupby("day_type")["occ_pct"].mean().round(2)
```

Out[104…
```
day_type
weekeday    50.88
weekend     72.34
Name: occ_pct, dtype: float64
```

In [105…
```
df["mmm yy"].unique()
```

Out[105…
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

4. In the month of June, What is the occupancy for different cities

In [107…
```
df_june_22 = df[df["mmm yy"]=="Jun 22"]
df_june_22.head(3)
```

Out[107…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	date
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury	Mumbai	10-Jun-22
2201	16559	10-Jun-22	RT2	26	41.0	63.41	Elite	Atliq Exotica	Luxury	Mumbai	10-Jun-22
2202	16559	10-Jun-22	RT3	20	32.0	62.50	Premium	Atliq Exotica	Luxury	Mumbai	10-Jun-22

In [109…
```
df_june_22.groupby("city")["occ_pct"].mean().round(2).sort_values(ascending= False)
```

Out[109…
```
city
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Bangalore    56.44
Name: occ_pct, dtype: float64
```

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

In [110…
```
df_august = pd.read_csv(r"C:\Users\HP\Projects - Python\new_data_august.csv")
df_august
```

Out[110…

	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
5	17561	Atliq Blu	Luxury	Mumbai	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	18
6	17564	Atliq Seasons	Business	Mumbai	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	10

In [111…
```
df_august.columns
```

Out[111…
```
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 [112…
```
df.columns
```

Out[112…
```
Index(['property_id', 'check_in_date', 'room_category', 'successful_bookings',
       'capacity', 'occ_pct', 'room_class', 'property_name', 'category',
       'city', 'date', 'mmm yy', 'week no', 'day_type'],
      dtype='object')
```

In [113…
```
df_august.shape
```

Out[113… (7, 13)

In [114…
```
df.shape
```

Out[114… (6497, 14)

In [115…
```
latest_df = pd.concat([df, df_august], ignore_index= True, axis = 0)
latest_df.tail(5)
```

Out[115…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	da
6499	19563	01-Aug-22	RT1	23	30.0	NaN	Standard	Atliq Palace	Business	Bangalore	Na
6500	19558	01-Aug-22	RT1	30	40.0	NaN	Standard	Atliq Grands	Luxury	Bangalore	Na
6501	19560	01-Aug-22	RT1	20	26.0	NaN	Standard	Atliq City	Business	Bangalore	Na
6502	17561	01-Aug-22	RT1	18	26.0	NaN	Standard	Atliq Blu	Luxury	Mumbai	Na
6503	17564	01-Aug-22	RT1	10	16.0	NaN	Standard	Atliq Seasons	Business	Mumbai	Na

In [117…
```
latest_df.shape
```

Out[117… (6504, 15)

6. Print revenue realized per day

In [118…
```
df_bookings.head()
```

Out [118…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
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	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	

In [119…

```
df_hotels.head()
```

Out [119…

	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 [120…

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

Out [120…

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

3	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0
4	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN

In [121…
```
df_bookings_all.groupby("city")["revenue_realized"].sum().round(2).sort_values(ascending=False)
```

Out[121…
```
city
Mumbai       668602231
Bangalore    420383550
Hyderabad    325179310
Delhi        294438788
Name: revenue_realized, dtype: int64
```

7. Print month by month revenue

In [122…
```
df_bookings_all.head(4)
```

Out[122…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	
1	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	5.0	
2	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	
3	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	

In [124…
```
df_date.head(3)
```

Out[124…

	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 [125…
```
df_date["mmm yy"].unique()
```

Out[125…
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

In [126…
```
pd.merge(df_bookings_all, df_date, left_on= "check_in_date", right_on="date") #here we didn't get o/p because date
```

Out[126…

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat

In [127…
```
df_bookings_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134578 entries, 0 to 134577
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   booking_id         134578 non-null  object 
 1   property_id        134578 non-null  int64  
 2   booking_date       134578 non-null  object 
 3   check_in_date      134578 non-null  object 
 4   checkout_date      134578 non-null  object 
 5   no_guests          134578 non-null  float64
 6   room_category      134578 non-null  object 
 7   booking_platform   134578 non-null  object 
 8   ratings_given      56679 non-null   float64
 9   booking_status     134578 non-null  object 
 10  revenue_generated  134578 non-null  int64  
 11  revenue_realized   134578 non-null  int64  
 12  property_name      134578 non-null  object 
 13  category           134578 non-null  object 
 14  city               134578 non-null  object 
dtypes: float64(2), int64(3), object(10)
memory usage: 15.4+ MB
```

In [128…
```
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 [129…
```
# Now we need to convert correct data format like (in both df's date is in object format we need to convert into dat
df_date["date"] = pd.to_datetime(df_date["date"])
df_date.head(3)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_13656\336666208.py:2: 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 spe
cify a format.
  df_date["date"] = pd.to_datetime(df_date["date"])
```

Out[129…

	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 [130…
```
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 [139…
```
df_bookings_all["check_in_date"] = pd.to_datetime(df_bookings_all["check_in_date"], format='mixed', dayfirst=True)
df_bookings_all.head(3)
```

Out[139…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
0	May012216558RT12	16558	30-04-22	2022-05-01	2/5/2022	2.0	RT1	others	NaN	
1	May012216558RT13	16558	28-04-22	2022-05-01	4/5/2022	2.0	RT1	logtrip	5.0	
2	May012216558RT15	16558	27-04-22	2022-05-01	2/5/2022	4.0	RT1	direct online	5.0	

In [135…
```
df_bookings_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134578 entries, 0 to 134577
Data columns (total 15 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   booking_id          134578 non-null  object 
 1   property_id         134578 non-null  int64  
 2   booking_date        134578 non-null  object 
 3   check_in_date       134578 non-null  object 
 4   checkout_date       134578 non-null  object 
 5   no_guests           134578 non-null  float64
 6   room_category       134578 non-null  object 
 7   booking_platform    134578 non-null  object 
 8   ratings_given       56679 non-null   float64
 9   booking_status      134578 non-null  object 
 10  revenue_generated   134578 non-null  int64  
 11  revenue_realized    134578 non-null  int64  
 12  property_name       134578 non-null  object 
 13  category            134578 non-null  object 
 14  city                134578 non-null  object 
dtypes: float64(2), int64(3), object(10)
memory usage: 15.4+ MB
```

In [140…
```
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on = "check_in_date", right_on = "date")
df_bookings_all.head(3)
```

Out [140…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
0	May012216558RT12	16558	30-04-22	2022-05-01	2/5/2022	2.0	RT1	others	NaN	
1	May012216558RT13	16558	28-04-22	2022-05-01	4/5/2022	2.0	RT1	logtrip	5.0	
2	May012216558RT15	16558	27-04-22	2022-05-01	2/5/2022	4.0	RT1	direct online	5.0	

In [144…
```
df_bookings_all[df_bookings_all["booking_id"] == "May052216558RT11"]
```

Out [144…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given
6159	May052216558RT11	16558	15-04-22	2022-05-05	7/5/2022	3.0	RT1	tripster	5.0

In [ ]:
```
#Print revenue realized per hotel type
df_bookings_all.groupby("property_name")["revenue_realized"].sum().sort_values(ascending=False)
```

Out[ ]:
```
property_name
Atliq Exotica    320291568
Atliq Palace     304081863
Atliq City       285798439
Atliq Blu        260851922
Atliq Bay        260022118
Atliq Grands     211471234
Atliq Seasons     66086735
Name: revenue_realized, dtype: int64
```

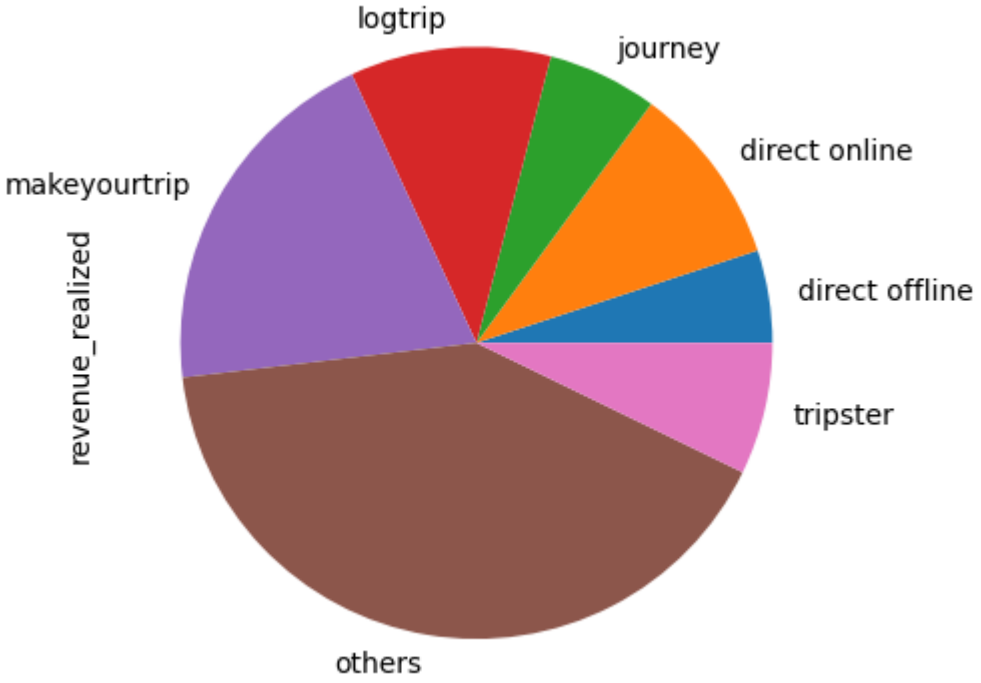
In [146…
```
# Print average rating per city
df_bookings_all.groupby("city")["ratings_given"].mean().round(2).sort_values(ascending= False)
```

Out [146…

```
city
Delhi        3.78
Hyderabad    3.66
Mumbai       3.65
Bangalore    3.41
Name: ratings_given, dtype: float64
```

In [148…

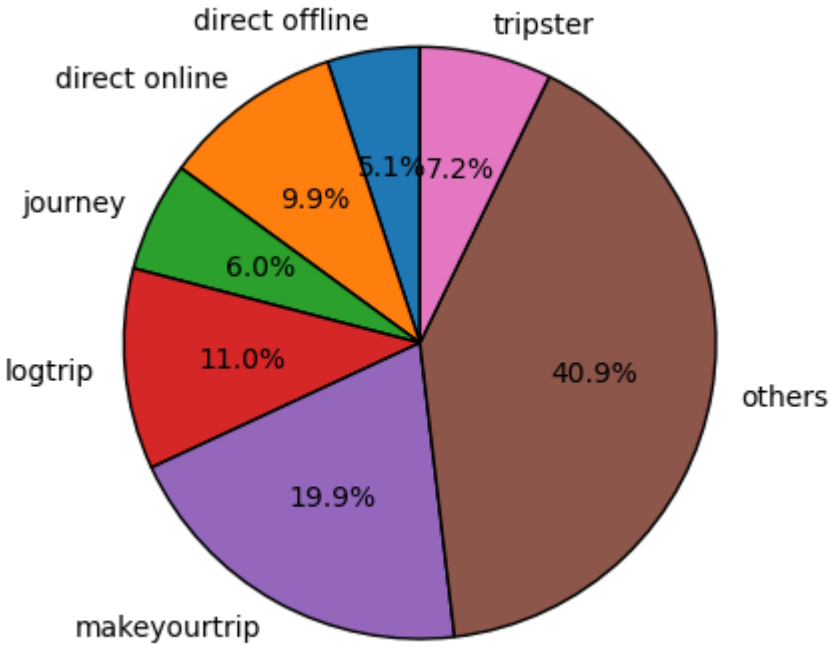
```
# Print a pie chart of revenue realized per booking platform
df_bookings_all.groupby("booking_platform")["revenue_realized"].sum().plot(kind = "pie")
plt.xticks(rotation = 0)
plt.show()
```



In [ ]:

```
df_bookings_all.groupby("booking_platform")["revenue_realized"].sum().plot(
    kind= "pie",
    autopct ="%1.1f%%", # Display the percentages with 1 decimal place
    startangle =90,     # Rotate the pie chart for better visibility
    wedgeprops ={'edgecolor': 'black'} # Add black border for better readability
```

```
)

plt.ylabel("")
plt.show()
```



In [ ]: