In [146…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df_bookings= pd.read_csv("fact_bookings.csv")
df_date = pd.read_csv('dim_date.csv')
df_hotels = pd.read_csv('dim_hotels.csv')
df_rooms = pd.read_csv('dim_rooms.csv')
df_agg_bookings = pd.read_csv('fact_aggregated_bookings.csv')
```

In [3]:
```
import warnings

# Ignore all warnings within this block
with warnings.catch_warnings():
    warnings.simplefilter("ignore")
```

Basic Data Exploration

In [4]:
```
df_bookings.head(6)
```

Out[4]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	bo
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	

In [5]:
```
df_bookings.shape
```

Out[5]:
```
(134590, 12)
```

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

Out[6]:
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

In [7]:
```
df_bookings.booking_platform.unique()
```

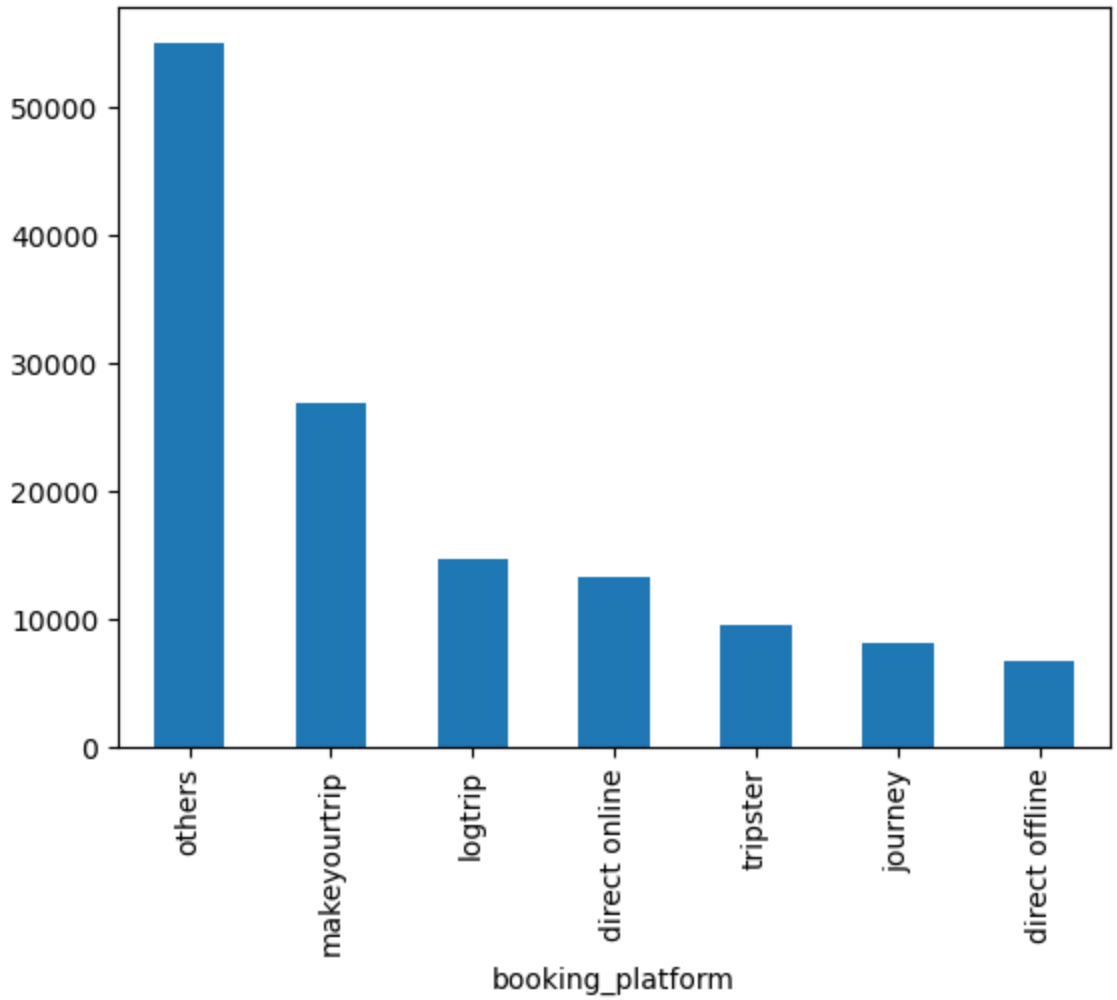
Out[7]:
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

In [8]:
```
df_bookings.booking_platform.value_counts()
```

Out[8]:
```
booking_platform
others            55066
makeyourtrip      26898
logtrip           14756
direct online     13379
tripster           9630
journey            8106
direct offline     6755
Name: count, dtype: int64
```

In [9]:
```
df_bookings.booking_platform.value_counts().plot(kind='bar')
```

Out[9]: <Axes: xlabel='booking_platform'>



In [10]:
```
df_bookings.describe(include= 'all')
```

Out[10]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_c
count	134590	134590.000000	134590	134590	134590	134587.000000	
unique	134590	NaN	116	92	97	NaN	
top	May012216558RT11	NaN	8/6/2022	16-07-22	9/5/2022	NaN	
freq	1	NaN	1670	2017	1840	NaN	
mean	NaN	18061.113493	NaN	NaN	NaN	2.036170	
std	NaN	1093.055847	NaN	NaN	NaN	1.034885	
min	NaN	16558.000000	NaN	NaN	NaN	-17.000000	
25%	NaN	17558.000000	NaN	NaN	NaN	1.000000	
50%	NaN	17564.000000	NaN	NaN	NaN	2.000000	
75%	NaN	18563.000000	NaN	NaN	NaN	2.000000	
max	NaN	19563.000000	NaN	NaN	NaN	6.000000	

In [11]:
```
df_bookings.revenue_generated.min(),df_bookings.revenue_generated.max()
```

Out[11]: (6500, 28560000)

In [12]:
```
df_date.head()
```

Out[12]:

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

Out[13]:

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

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

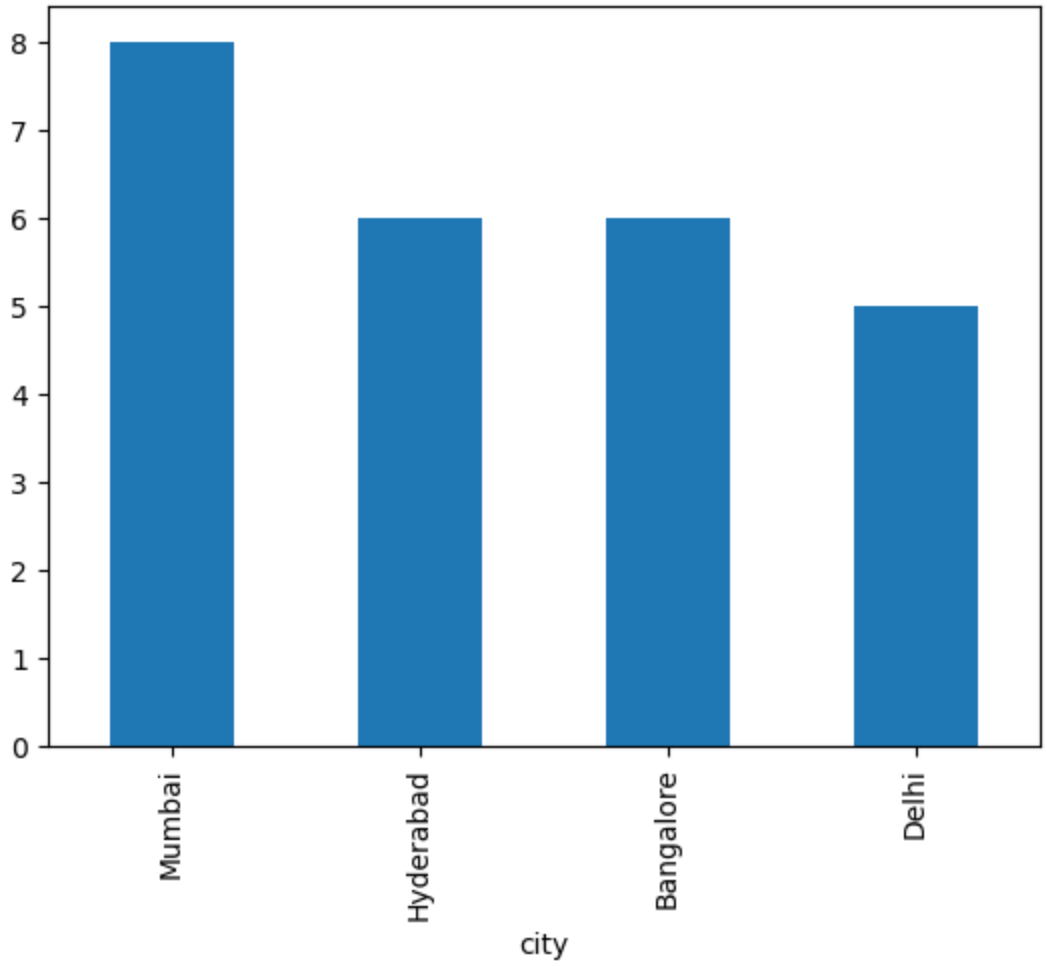
Out[14]:
```
category
Luxury      16
Business     9
Name: count, dtype: int64
```

In [15]:
```
df_hotels.city.value_counts().sort_values(ascending=False) ## Default ASC
```

Out[15]:
```
city
Mumbai       8
Hyderabad    6
Bangalore    6
Delhi        5
Name: count, dtype: int64
```

In [16]:
```
df_hotels.city.value_counts().sort_values(ascending=False).plot(kind = 'bar')
```

Out[16]:
```
<Axes: xlabel='city'>
```



In [17]:
```
df_rooms.head()
```

Out[17]:

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

In [18]:
```
df_agg_bookings.head()
```

Out[18]:

	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

1. Find out unique property ids in aggregate bookings dataset

In [19]:
```
df_agg_bookings.property_id.unique()
```

Out[19]:
```
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)
```

2. Find out total bookings per property_id

In [20]:
```
df_agg_bookings.property_id.value_counts()
```

Out[20]:
```
property_id
16559    368
17559    368
17564    368
19561    368
19559    368
18563    368
18562    368
18561    368
18559    368
18558    368
17563    368
17562    368
16563    368
19562    368
16562    368
16561    368
16560    368
17561    368
19560    368
19558    368
17560    368
16558    368
17558    368
19563    368
18560    368
Name: count, dtype: int64
```

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

In [21]:
```
## Accessing columns

df_agg_bookings.loc[:,'capacity']
df_agg_bookings.capacity
df_agg_bookings['capacity']
```

Out[21]:
```
0       30.0
1       30.0
2       30.0
3       19.0
4       19.0
        ...
9195    18.0
9196    18.0
9197     6.0
9198     6.0
9199     4.0
Name: capacity, Length: 9200, dtype: float64
```

In [22]:
```
## Accessing 2 columns

df_agg_bookings[['successful_bookings','capacity']]

df_agg_bookings.loc[:,['successful_bookings','capacity']]
```

Out[22]:

	successful_bookings	capacity
0	25	30.0
1	28	30.0
2	23	30.0
3	30	19.0
4	18	19.0
...	...	...
9195	13	18.0
9196	13	18.0
9197	3	6.0
9198	3	6.0
9199	3	4.0

9200 rows × 2 columns

In [23]:
```
#Accessing all the columns in dataframe using specific condition

df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.capacity]

#Accessing only 2 specific columns in dataframe using specific condition

df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.capacity] [['success
```

Out[23]:

	successful_bookings	capacity
3	30	19.0
12	100	41.0
4136	50	39.0
6209	123	26.0
8522	35	24.0
9194	20	18.0

In [24]:
```
df_agg_bookings.query('successful_bookings > capacity') [['successful_bookings','capacit
```

Out[24]:

	successful_bookings	capacity
3	30	19.0
12	100	41.0
4136	50	39.0
6209	123	26.0
8522	35	24.0
9194	20	18.0

4. Find out properties that have highest capacity

In [25]:
```
df_agg_bookings.loc[:,'capacity'].max()
```

Out[25]:
```
50.0
```

In [26]:
```
df_agg_bookings.capacity.max()
```

Out[26]:
```
50.0
```

In [27]:
```
df_agg_bookings
```

Out[27]:

	property_id	check_in_date	room_category	successful_bookings	capacity
0	16559	1-May-22	RT1	25	30.0
1	19562	1-May-22	RT1	28	30.0
2	19563	1-May-22	RT1	23	30.0
3	17558	1-May-22	RT1	30	19.0
4	16558	1-May-22	RT1	18	19.0
...	...	...	...	...	...
9195	16563	31-Jul-22	RT4	13	18.0
9196	16559	31-Jul-22	RT4	13	18.0
9197	17558	31-Jul-22	RT4	3	6.0
9198	19563	31-Jul-22	RT4	3	6.0
9199	17561	31-Jul-22	RT4	3	4.0

9200 rows × 5 columns

In [28]:
```
p = df_agg_bookings[df_agg_bookings.capacity == df_agg_bookings.capacity.max()] [['prope

p
```

Out[28]:

	property_id	capacity	room_category
27	17558	50.0	RT2
128	17558	50.0	RT2
229	17558	50.0	RT2
328	17558	50.0	RT2
428	17558	50.0	RT2
...	...	...	...
8728	17558	50.0	RT2
8828	17558	50.0	RT2
8928	17558	50.0	RT2
9028	17558	50.0	RT2
9128	17558	50.0	RT2

92 rows × 3 columns

In [29]:
```
p.room_category.unique()
```

Out[29]:
```
array(['RT2'], dtype=object)
```

==> 2. Data Cleaning

In [30]:
```
df_bookings
```

Out[30]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_categor
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT
...	...	...	...	...	...	...	...
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	1.0	RT
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	-4.0	RT
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	1.0	RT
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	2.0	RT
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	2.0	RT

134590 rows × 12 columns

Removing the data from data frame with negative guests

In [31]:
```
df_bookings[df_bookings.no_guests<0]
```

Out[31]:

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

In [32]:
```
### Removing the data from data frame with negative guests

df_bookings = df_bookings[df_bookings.no_guests>0]

df_bookings
```

Out[32]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_categor
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT
...	...	...	...	...	...	...	.
134584	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	2.0	RT
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	1.0	RT
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	1.0	RT
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	2.0	RT
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	2.0	RT

134578 rows × 12 columns

(2) Outlier removal in revenue generated*

In [33]:
```
df_bookings.revenue_generated.describe()
```

Out[33]:
```
count    1.345780e+05
mean     1.537804e+04
std      9.304015e+04
min      6.500000e+03
25%      9.900000e+03
50%      1.350000e+04
75%      1.800000e+04
max      2.856000e+07
Name: revenue_generated, dtype: float64
```

In [34]:
```
df_bookings.revenue_generated.min(),df_bookings.revenue_generated.max(),df_bookings.reve
```

Out[34]:
```
(6500, 28560000, 15378.036937686695)
```

In [35]:
```
lower_limit = df_bookings.revenue_generated.mean() - 3*df_bookings.revenue_generated.std
lower_limit
```

Out[35]:
```
-263742.4278566132
```

In [36]:
```
higher_limit = df_bookings.revenue_generated.mean() + 3*df_bookings.revenue_generated.st
higher_limit
```

Out[36]:
```
294498.50173198653
```

In [37]:
```
df_bookings[df_bookings.revenue_generated > higher_limit]
```

Out[37]:

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

In [38]:
```
df_bookings = df_bookings[df_bookings.revenue_generated < higher_limit]

df_bookings
```

Out[38]:

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

134573 rows × 12 columns

(2) Outlier removal in revenue generated

In [39]:
```
df_bookings.revenue_realized.describe()
```

Out[39]:
```
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 [40]:
```
lower_limit = df_bookings.revenue_realized.mean() - 3*df_bookings.revenue_realized.std()
lower_limit
```

Out[40]:
```
-8087.391491610155
```

In [41]:
```
higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std(
higher_limit
```

Out[41]:
```
33479.3586618449
```

In [42]:
```
df_bookings[df_bookings.revenue_realized>higher_limit].shape
```

Out[42]: (1299, 12)

In [43]:
```
df_bookings[df_bookings.revenue_realized>higher_limit]
```

Out[43]:

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

1299 rows × 12 columns

In [44]:
```
df_bookings[df_bookings.revenue_realized>higher_limit].room_category.value_counts()
```

Out[44]:
```
room_category
RT4    1299
Name: count, dtype: int64
```

In [45]:
```
df_bookings[df_bookings.room_category == 'RT4'].revenue_realized.describe()
```

Out[45]:
```
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 [46]:
```
lower_limit = 23439.308444 - 3*9048.599076
lower_limit
```

Out[46]:
```
-3706.4887840000047
```

In [47]:
```
Higher_limit = 23439.308444 + 3*9048.599076
Higher_limit
```

Out[47]:
```
50585.105672000005
```

In [48]:
```
df_bookings.isnull().sum()
```

Out[48]:
```
booking_id                0
property_id               0
booking_date              0
check_in_date             0
checkout_date             0
no_guests                 0
room_category             0
booking_platform          0
ratings_given         77897
booking_status            0
revenue_generated         0
revenue_realized          0
dtype: int64
```

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

In [49]:
```
df_agg_bookings.isnull().sum()
```

Out[49]:
```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               2
dtype: int64
```

In [50]:
```
M = df_agg_bookings.capacity.mean()
M
```

Out[50]:
```
25.280495759947815
```

In [51]:
```
df_agg_bookings['capacity'].fillna(25.280495759947815,inplace = True)
```

In [52]:
```
df_agg_bookings['capacity'].isnull().sum()
```

Out[52]:
```
0
```

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

In [53]:
```
df_agg_bookings.head()
```

Out[53]:

	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 [54]:
```
df_agg_bookings[df_agg_bookings.successful_bookings > df_agg_bookings.capacity]
```

Out[54]:

	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 [55]:
```
df_agg_bookings = df_agg_bookings[df_agg_bookings.successful_bookings <=df_agg_bookings.
```

In [56]:
```
df_agg_bookings.shape
```

Out[56]:
```
(9194, 5)
```

==> 3. Data Transformation

Create occupancy percentage column

In [57]:
```
df_agg_bookings.head()
```

Out[57]:

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

In [58]:
```
df_agg_bookings['occu_per']= round((df_agg_bookings['successful_bookings'] / df_agg_book

## df_agg_bookings['occu_per'] = df_agg_bookings['occu_per'].apply(lambda x: round(x*100
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_6968\2025026760.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df_agg_bookings['occu_per']= round((df_agg_bookings['successful_bookings'] / df_agg_bo
okings['capacity'])*100,2)
```

In [59]:
```
df_agg_bookings
```

Out[59]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occu_per
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
...	...	...	...	...	...	...
9195	16563	31-Jul-22	RT4	13	18.0	72.22
9196	16559	31-Jul-22	RT4	13	18.0	72.22
9197	17558	31-Jul-22	RT4	3	6.0	50.00
9198	19563	31-Jul-22	RT4	3	6.0	50.00
9199	17561	31-Jul-22	RT4	3	4.0	75.00

9194 rows × 6 columns

In [60]:
```
df_agg_bookings.head()
```

Out[60]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occu_per
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

==> 4. Insights Generation

---

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

In [61]:
```
G = df_agg_bookings.groupby('room_category')
```

In [62]:
```
G.occu_per.mean()
```

Out[62]:
```
room_category
RT1    57.888985
RT2    58.009756
RT3    58.028213
RT4    59.277925
Name: occu_per, dtype: float64
```

In [63]:
```
df_agg_bookings.groupby('room_category').occu_per.mean()
```

Out[63]:
```
room_category
RT1    57.888985
RT2    58.009756
RT3    58.028213
RT4    59.277925
Name: occu_per, dtype: float64
```

In [64]:
```
df_agg_bookings.groupby('room_category') ['occu_per'].mean()
```

Out[64]:
```
room_category
RT1    57.888985
RT2    58.009756
RT3    58.028213
RT4    59.277925
Name: occu_per, dtype: float64
```

In [65]:
```
df_rooms
```

Out[65]:

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

In [66]:
```
mer = pd.merge(df_agg_bookings,df_rooms,left_on='room_category', right_on='room_id')
```

In [67]:
```
mer
```

Out[67]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occu_per	room_id	room_clas
0	16559	1-May-22	RT1	25	30.0	83.33	RT1	Standar
1	19562	1-May-22	RT1	28	30.0	93.33	RT1	Standar
2	19563	1-May-22	RT1	23	30.0	76.67	RT1	Standar
3	16558	1-May-22	RT1	18	19.0	94.74	RT1	Standar
4	17560	1-May-22	RT1	28	40.0	70.00	RT1	Standar
...	...	...	...	...	...	...	...	.
9189	16563	31-Jul-22	RT4	13	18.0	72.22	RT4	Presidentia
9190	16559	31-Jul-22	RT4	13	18.0	72.22	RT4	Presidentia
9191	17558	31-Jul-22	RT4	3	6.0	50.00	RT4	Presidentia
9192	19563	31-Jul-22	RT4	3	6.0	50.00	RT4	Presidentia
9193	17561	31-Jul-22	RT4	3	4.0	75.00	RT4	Presidentia

9194 rows × 8 columns

In [68]:
```
mer.groupby('room_class')['occu_per'].mean().round(2)
```

Out[68]:
```
room_class
Elite           58.01
Premium         58.03
Presidential    59.28
Standard        57.89
Name: occu_per, dtype: float64
```

In [69]:
```
mer.drop('room_id',axis = 1,inplace= True) ## Dropped room_id as we have 2 columns with
```

In [70]:
```
mer
```

Out[70]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occu_per	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
...	...	...	...	...	...	...	...
9189	16563	31-Jul-22	RT4	13	18.0	72.22	Presidential
9190	16559	31-Jul-22	RT4	13	18.0	72.22	Presidential
9191	17558	31-Jul-22	RT4	3	6.0	50.00	Presidential
9192	19563	31-Jul-22	RT4	3	6.0	50.00	Presidential
9193	17561	31-Jul-22	RT4	3	4.0	75.00	Presidential

9194 rows × 7 columns

2. Print average occupancy rate per city

In [71]:
```
df_hotels.head()
```

Out[71]:

	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 [72]:
```
merg = pd.merge(mer,df_hotels,on='property_id',how = 'outer')
```

In [73]:
```
merg
```

Out[73]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occu_per	room_class	propert
0	16559	1-May-22	RT1	25	30.0	83.33	Standard	Atli
1	16559	2-May-22	RT1	20	30.0	66.67	Standard	Atli
2	16559	3-May-22	RT1	17	30.0	56.67	Standard	Atli
3	16559	4-May-22	RT1	21	30.0	70.00	Standard	Atli
4	16559	5-May-22	RT1	16	30.0	53.33	Standard	Atli
...	...	...	...	...	...	...	...	
9189	16563	27-Jul-22	RT4	10	18.0	55.56	Presidential	Atli
9190	16563	28-Jul-22	RT4	9	18.0	50.00	Presidential	Atli
9191	16563	29-Jul-22	RT4	9	18.0	50.00	Presidential	Atli
9192	16563	30-Jul-22	RT4	11	18.0	61.11	Presidential	Atli
9193	16563	31-Jul-22	RT4	13	18.0	72.22	Presidential	Atli

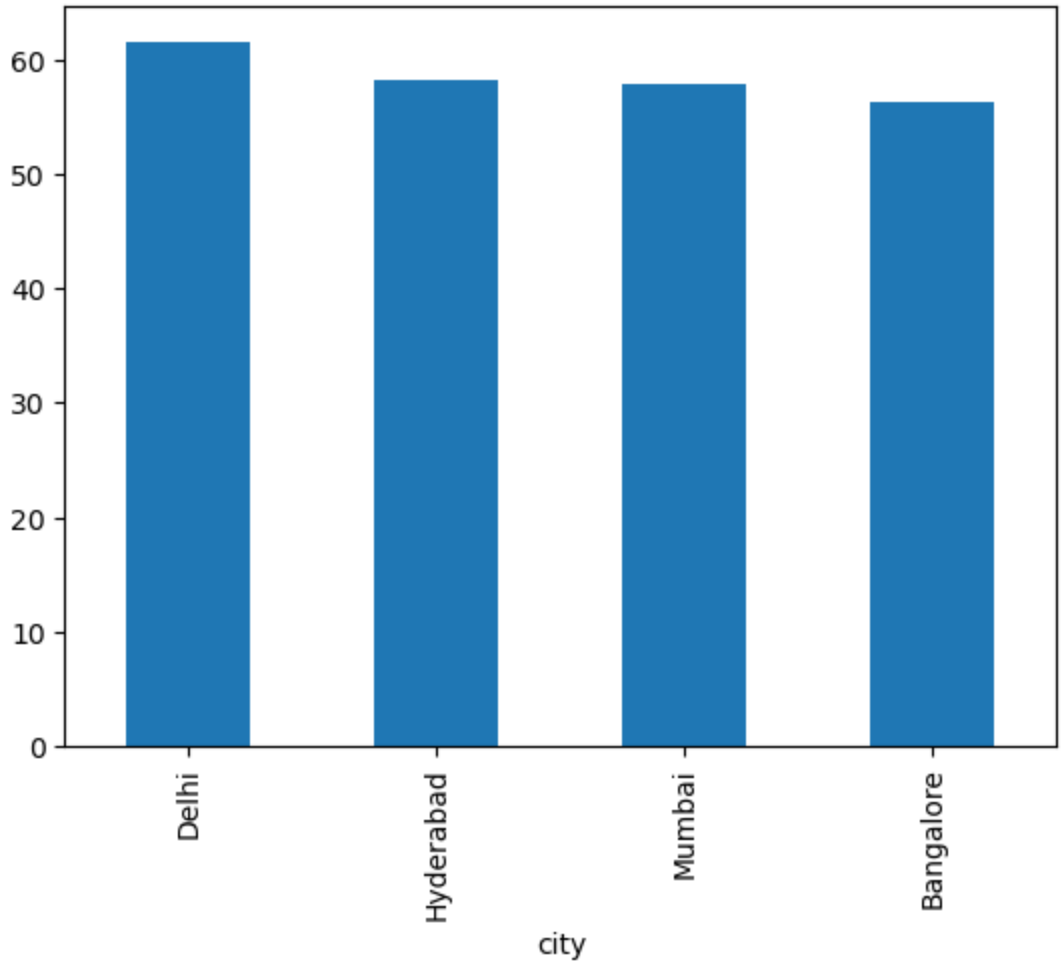
9194 rows × 10 columns

In [74]:
```
merg.groupby('city') ['occu_per'].mean().round(2)
```

Out[74]:
```
city
Bangalore    56.33
Delhi        61.51
Hyderabad    58.12
Mumbai       57.91
Name: occu_per, dtype: float64
```

In [75]:
```
merg.groupby('city') ['occu_per'].mean().round(2).sort_values(ascending= False).plot(kin
```

Out[75]:
```
<Axes: xlabel='city'>
```



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

In [76]:
```
merg.head()
```

Out[76]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occu_per	room_class	property_r
0	16559	1-May-22	RT1	25	30.0	83.33	Standard	Atliq Ex
1	16559	2-May-22	RT1	20	30.0	66.67	Standard	Atliq Ex
2	16559	3-May-22	RT1	17	30.0	56.67	Standard	Atliq Ex
3	16559	4-May-22	RT1	21	30.0	70.00	Standard	Atliq Ex
4	16559	5-May-22	RT1	16	30.0	53.33	Standard	Atliq Ex

In [77]:
```
df_date.head()
```

Out[77]:

	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 [78]:
```
merge_df = pd.merge(merg, df_date,left_on = 'check_in_date',right_on = 'date')
```

In [79]:
```
merge_df.head()
```

Out[79]:

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

In [80]:
```
merge_df.groupby('day_type') ['occu_per'].mean().round(2)
```

Out[80]:
```
day_type
weekeday    50.88
weekend     72.34
Name: occu_per, dtype: float64
```

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

In [81]:
```
merge_df.head()
```

Out[81]:

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

In [82]:
```
Jun = merge_df.groupby(['city','mmm yy']) ['occu_per'].mean()
Jun
```

Out[82]:
```
city       mmm yy
Bangalore  Jul 22    53.899829
           Jun 22    56.436143
           May 22    55.275492
Delhi      Jul 22    59.177886
           Jun 22    62.474286
           May 22    59.650614
Hyderabad  Jul 22    55.252163
           Jun 22    58.458075
           May 22    57.062405
Mumbai     Jul 22    55.235469
           Jun 22    58.382560
           May 22    56.803139
Name: occu_per, dtype: float64
```

In [83]:
```
Jun.loc[Jun.index.get_level_values('mmm yy') =='Jun 22',:]
```

Out[83]:
```
city       mmm yy
Bangalore  Jun 22    56.436143
Delhi      Jun 22    62.474286
Hyderabad  Jun 22    58.458075
Mumbai     Jun 22    58.382560
Name: occu_per, dtype: float64
```

Another approach

In [84]:
```
 Jun_data = merge_df[merge_df['mmm yy'] == 'Jun 22']
```

In [85]:
```
Jun_data.head()
```

Out[85]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occu_per	room_class	propert
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	Atli
2201	16559	10-Jun-22	RT2	26	41.0	63.41	Elite	Atli
2202	16559	10-Jun-22	RT3	20	32.0	62.50	Premium	Atli
2203	16559	10-Jun-22	RT4	11	18.0	61.11	Presidential	Atli
2204	19562	10-Jun-22	RT1	19	30.0	63.33	Standard	

In [86]:
```
Jun_data.groupby(['city','mmm yy'])['occu_per'].mean()
```

Out[86]:
```
city       mmm yy
Bangalore  Jun 22    56.436143
Delhi      Jun 22    62.474286
Hyderabad  Jun 22    58.458075
Mumbai     Jun 22    58.382560
Name: occu_per, dtype: float64
```

In [87]:
```
df_august = pd.read_csv("new_data_august.csv")
```

In [88]:
```
df_august.head()
```

Out[88]:

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

In [89]:
```
New_df = pd.concat([merge_df,df_august],ignore_index = True,axis = 0)
```

In [90]:
```
New_df.tail()
```

Out[90]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occu_per	room_class	propert
6499	19563	01-Aug-22	RT1	23	30.0	NaN	Standard	Atli
6500	19558	01-Aug-22	RT1	30	40.0	NaN	Standard	Atli
6501	19560	01-Aug-22	RT1	20	26.0	NaN	Standard	
6502	17561	01-Aug-22	RT1	18	26.0	NaN	Standard	
6503	17564	01-Aug-22	RT1	10	16.0	NaN	Standard	Atliq

In [91]:
```
New_df=New_df.drop('date',axis = 1)
```

In [92]:
```
New_df.tail()
```

Out[92]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occu_per	room_class	propert
6499	19563	01-Aug-22	RT1	23	30.0	NaN	Standard	Atli
6500	19558	01-Aug-22	RT1	30	40.0	NaN	Standard	Atli
6501	19560	01-Aug-22	RT1	20	26.0	NaN	Standard	
6502	17561	01-Aug-22	RT1	18	26.0	NaN	Standard	
6503	17564	01-Aug-22	RT1	10	16.0	NaN	Standard	Atliq

6. Print revenue realized per city

In [93]:
```
df_bookings.head()
```

Out[93]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	bo
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	

In [94]:
```
df_hotels.head()
```

Out[94]:

	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 [95]:
```
rev_realized = pd.merge(df_bookings,df_hotels,on = 'property_id')
```

In [96]:
```
rev_realized.head()
```

Out[96]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	bo
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	
3	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	
4	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	

In [97]:
```
rev_realized.groupby('city')['revenue_realized'].sum()
```

Out[97]:
```
city
Bangalore    420383550
Delhi        294404488
Hyderabad    325179310
Mumbai       668569251
Name: revenue_realized, dtype: int64
```

7. Print month by month revenue

In [98]:
```
df_date.head(3)
```

Out[98]:

	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 [99]:
```
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 [100…
```
df_bookings.head(3)
```

Out[100]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	b
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	

In [101…
```
df_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 134573 entries, 1 to 134589
Data columns (total 12 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  
dtypes: float64(2), int64(3), object(7)
memory usage: 13.3+ MB
```

In [ ]:
```
df_date['date']= pd.to_datetime(df_date['date'])

df_date.head(3)
```

In [ ]:
```
import pandas as pd
df_bookings['check_in_date'] = pd.to_datetime(df_bookings['check_in_date'], errors='coer
```

In [104…
```
df_bookings.head(3)
```

Out[104]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	b
1	May012216558RT12	16558	30-04-22	2022-01-05	2/5/2022	2.0	RT1	
4	May012216558RT15	16558	27-04-22	2022-01-05	2/5/2022	4.0	RT1	
5	May012216558RT16	16558	1/5/2022	2022-01-05	3/5/2022	2.0	RT1	

In [105…
```
df_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 134573 entries, 1 to 134589
Data columns (total 12 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      55790 non-null   datetime64[ns]
 4   checkout_date      134573 non-null  object        
 5   no_guests          134573 non-null  float64       
 6   room_category      134573 non-null  object        
 7   booking_platform   134573 non-null  object        
 8   ratings_given      56676 non-null   float64       
 9   booking_status     134573 non-null  object        
 10  revenue_generated  134573 non-null  int64         
 11  revenue_realized   134573 non-null  int64         
dtypes: datetime64[ns](1), float64(2), int64(3), object(6)
memory usage: 13.3+ MB
```

In [107…
```
df_bookings_all = pd.merge(df_bookings, df_date, left_on="check_in_date", right_on="date
df_bookings_all.head(3)
```

Out[107]:

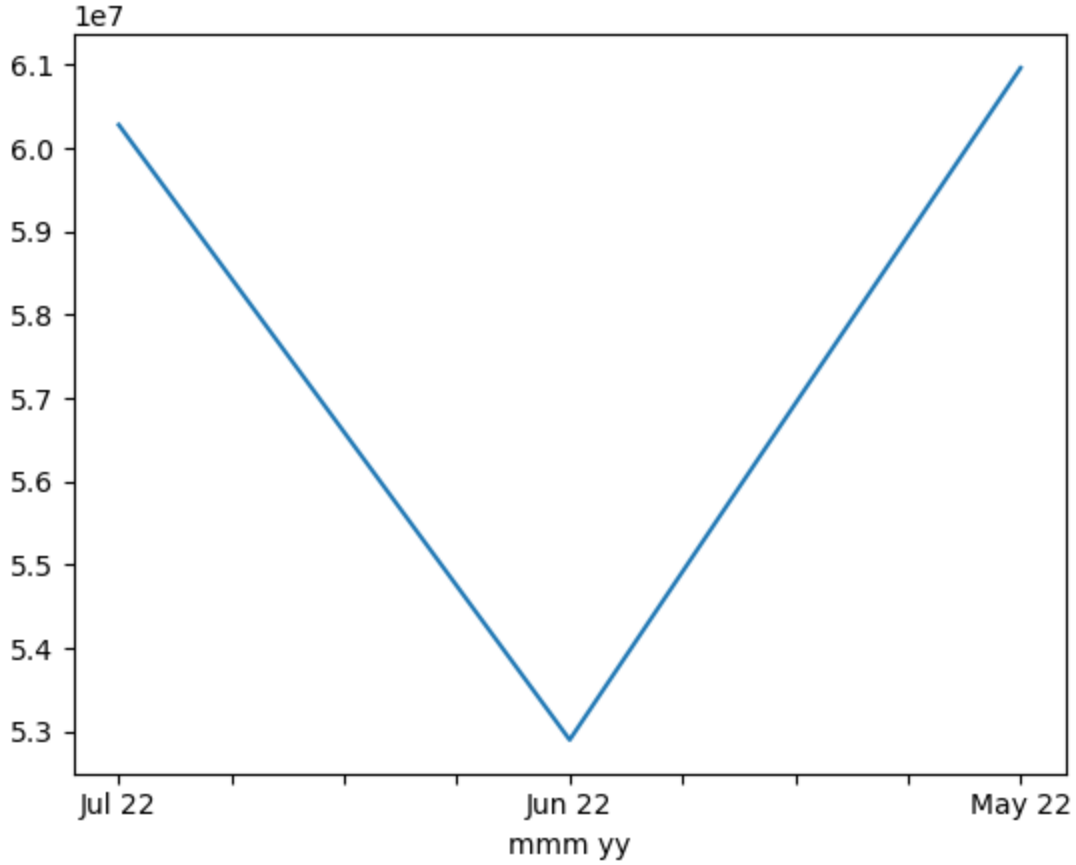
	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	b
0	May052216558RT11	16558	15-04-22	2022-05-05	7/5/2022	3.0	RT1	
1	May052216558RT12	16558	30-04-22	2022-05-05	7/5/2022	2.0	RT1	
2	May052216558RT13	16558	1/5/2022	2022-05-05	6/5/2022	3.0	RT1	

In [135…
```
df_bookings_all.groupby('mmm yy')['revenue_realized'].sum()
```

Out[135]:
```
mmm yy
Jul 22    60278496
Jun 22    52903014
May 22    60961428
Name: revenue_realized, dtype: int64
```

In [134…
```
df_bookings_all.groupby('mmm yy')['revenue_realized'].sum().plot()
```

Out[134]:
```
<Axes: xlabel='mmm yy'>
```



Exercise-1. Print revenue realized per hotel type

In [114…
```
df_hotels.head(3)
```

Out[114]:

	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 [116…
```
df_bookings.head(3)
```

Out[116]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	b
1	May012216558RT12	16558	30-04-22	2022-01-05	2/5/2022	2.0	RT1	
4	May012216558RT15	16558	27-04-22	2022-01-05	2/5/2022	4.0	RT1	
5	May012216558RT16	16558	1/5/2022	2022-01-05	3/5/2022	2.0	RT1	

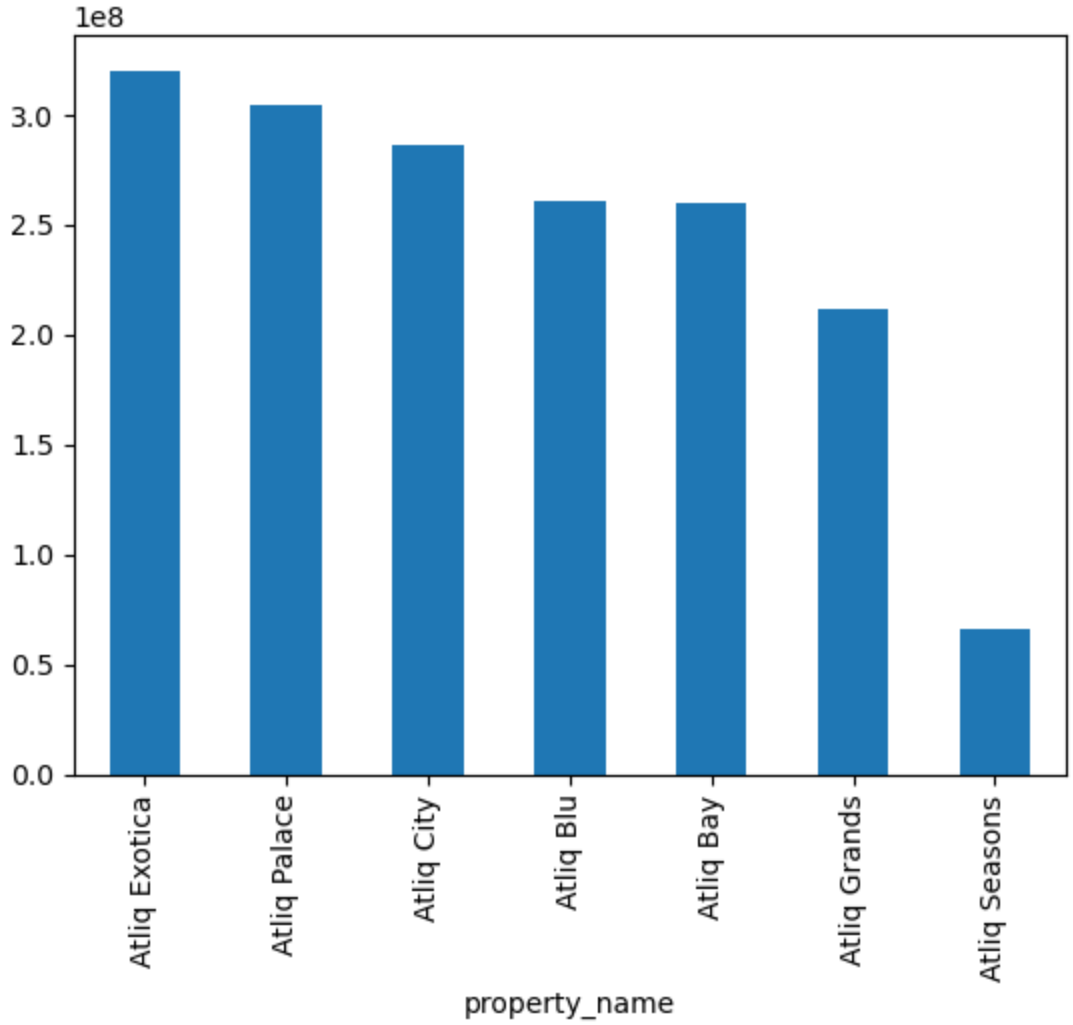
In [127…
```
hotels_data = pd.merge(df_hotels,df_bookings,on= 'property_id')
hotels_data.head(3)
```

Out[127]:

	property_id	property_name	category	city	booking_id	booking_date	check_in_date	checkout_dat
0	16558	Atliq Grands	Luxury	Delhi	May012216558RT12	30-04-22	2022-01-05	2/5/202
1	16558	Atliq Grands	Luxury	Delhi	May012216558RT15	27-04-22	2022-01-05	2/5/202
2	16558	Atliq Grands	Luxury	Delhi	May012216558RT16	1/5/2022	2022-01-05	3/5/202

In [128…
```
hotels_data.groupby('property_name')['revenue_realized'].sum().sort_values(ascending = F
```

Out[128]: <Axes: xlabel='property_name'>



Exercise-2 Print average rating per city

In [129…
```
hotels_data.head(3)
```

Out[129]:

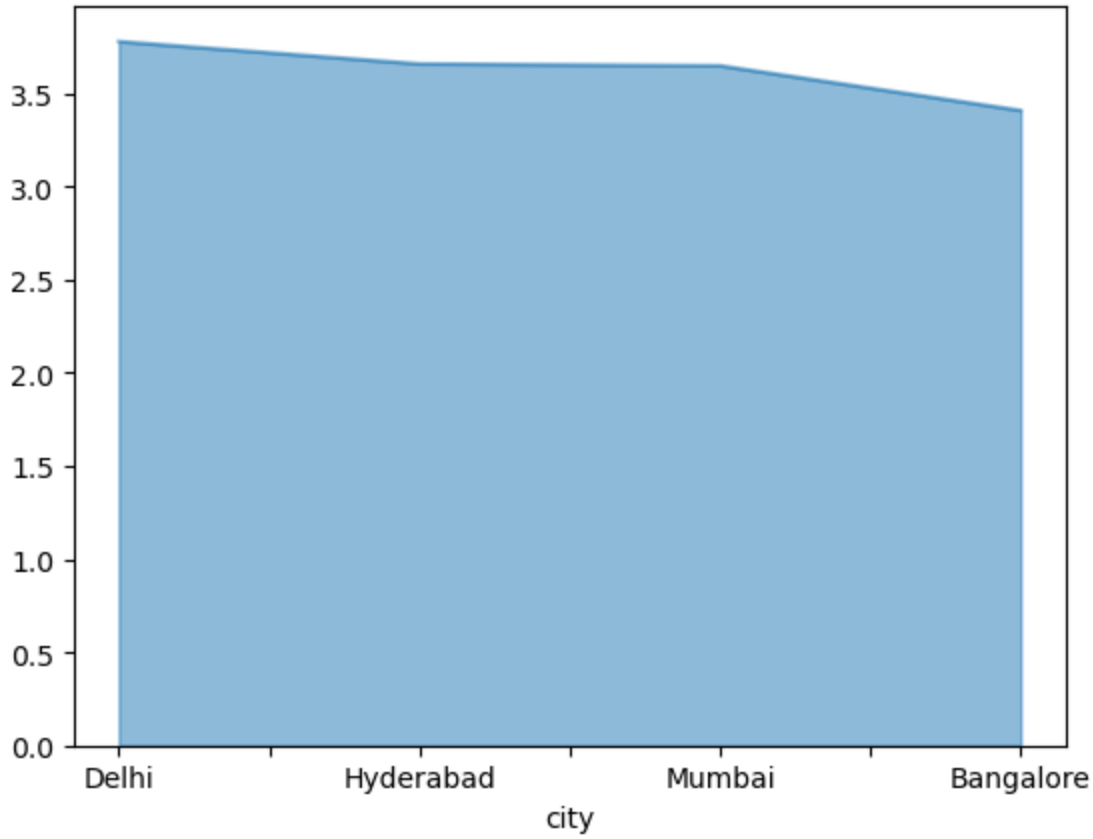
	property_id	property_name	category	city	booking_id	booking_date	check_in_date	checkout_dat
0	16558	Atliq Grands	Luxury	Delhi	May012216558RT12	30-04-22	2022-01-05	2/5/202
1	16558	Atliq Grands	Luxury	Delhi	May012216558RT15	27-04-22	2022-01-05	2/5/202
2	16558	Atliq Grands	Luxury	Delhi	May012216558RT16	1/5/2022	2022-01-05	3/5/202

In [138…
```
hotels_data.groupby('city')['ratings_given'].mean().round(2).sort_values(ascending = Fal
```

Out[138]:
```
city
Delhi        3.78
Hyderabad    3.66
Mumbai       3.65
Bangalore    3.41
Name: ratings_given, dtype: float64
```

In [142…
```
hotels_data.groupby('city')['ratings_given'].mean().round(2).sort_values(ascending = Fal
```

Out[142]: <Axes: xlabel='city'>



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

In [143…
```
hotels_data.head(3)
```

Out[143]:

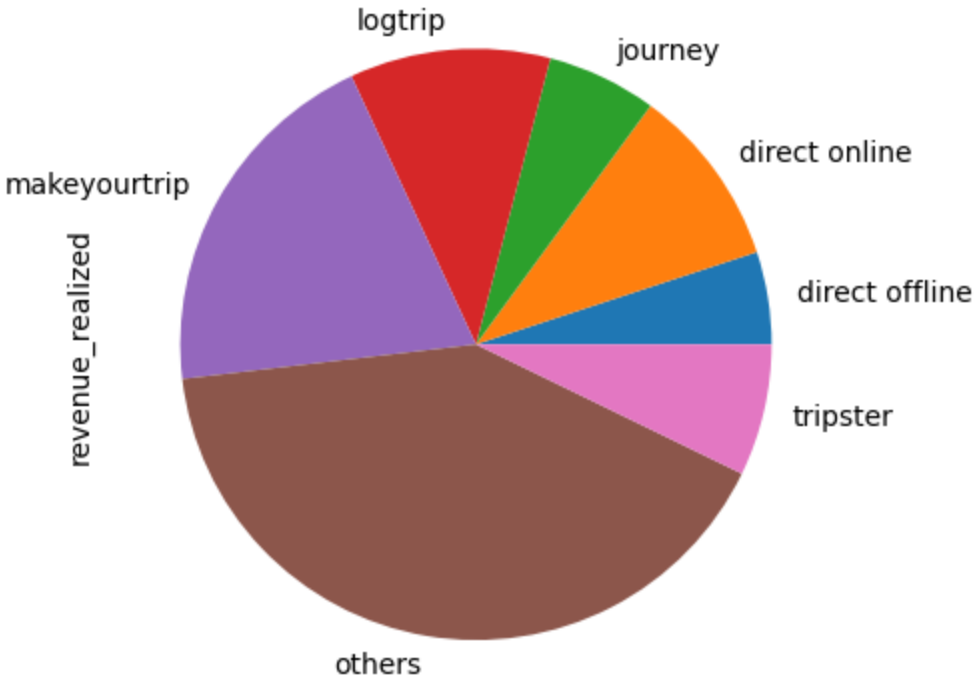
	property_id	property_name	category	city	booking_id	booking_date	check_in_date	checkout_dat
0	16558	Atliq Grands	Luxury	Delhi	May012216558RT12	30-04-22	2022-01-05	2/5/202
1	16558	Atliq Grands	Luxury	Delhi	May012216558RT15	27-04-22	2022-01-05	2/5/202
2	16558	Atliq Grands	Luxury	Delhi	May012216558RT16	1/5/2022	2022-01-05	3/5/202

In [144…
```
hotels_data.groupby('booking_platform')['revenue_realized'].sum()
```

Out[144]:
```
booking_platform
direct offline     86374933
direct online     168948637
journey           102531334
logtrip           187494028
makeyourtrip      340814104
others            699306762
tripster          123066801
Name: revenue_realized, dtype: int64
```

In [148…
```
hotels_data.groupby('booking_platform')['revenue_realized'].sum().plot(kind = 'pie')
```

Out[148]:
```
<Axes: ylabel='revenue_realized'>
```



In [ ]: