AtliQ Hotels Data Analysis Project

¶

In [20]:
```
import pandas as pd
```

==> 1. Data Import and Data Exploration

Datasets

We have 5 csv file

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

Read bookings data in a datagrame

In [21]:
```
df_bookings = pd.read_csv('datasets/fact_bookings.csv')
```

Explore bookings data

In [22]:
```
df_bookings.head()
```

Out[22]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	1.0	Checked C
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancell
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	5.0	Checked C
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	NaN	Cancell
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	Checked C

In [23]:
```
df_bookings.shape
```

Out[23]:
```
(134590, 12)
```

In [24]:
```
df_bookings.room_category.unique()
```

Out[24]:
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

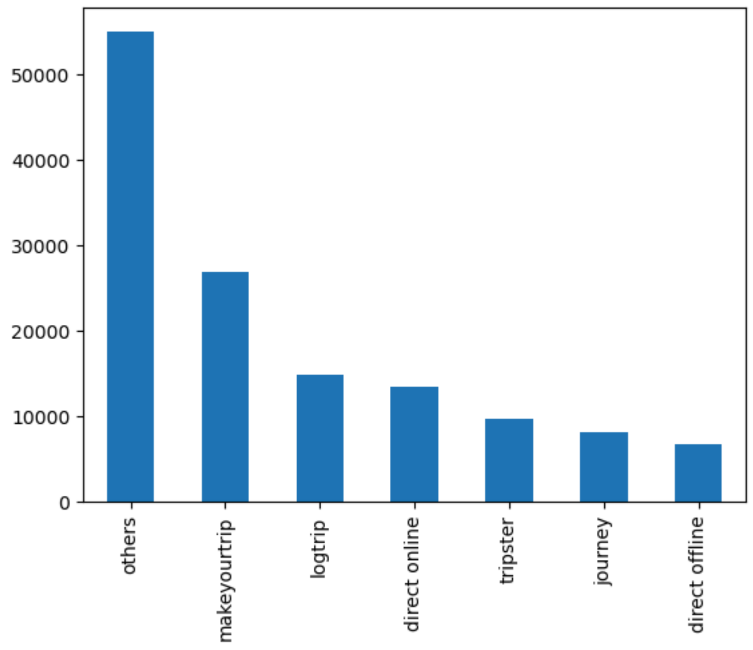
In [25]:
```
df_bookings.booking_platform.unique()
```

Out[25]:
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

In [26]:
```
df_bookings.booking_platform.value_counts()
```

Out[26]:
```
others            55066
makeyourtrip      26898
logtrip           14756
direct online     13379
tripster           9630
journey            8106
direct offline     6755
Name: booking_platform, dtype: int64
```

```
In [27]: df_bookings.booking_platform.value_counts().plot(kind="bar")
Out[27]: <Axes: >
```



```
In [28]: df_bookings.describe()
Out[28]:
```

	property_id	no_guests	ratings_given	revenue_generated	revenue_realized
count	134590.000000	134587.000000	56683.000000	1.345900e+05	134590.000000
mean	18061.113493	2.036170	3.619004	1.537805e+04	12696.123256
std	1093.055847	1.034885	1.235009	9.303604e+04	6928.108124
min	16558.000000	-17.000000	1.000000	6.500000e+03	2600.000000
25%	17558.000000	1.000000	3.000000	9.900000e+03	7600.000000
50%	17564.000000	2.000000	4.000000	1.350000e+04	11700.000000
75%	18563.000000	2.000000	5.000000	1.800000e+04	15300.000000
max	19563.000000	6.000000	5.000000	2.856000e+07	45220.000000

Read rest of the files

```
In [29]: df_date = pd.read_csv('datasets/dim_date.csv')
         df_hotels = pd.read_csv('datasets/dim_hotels.csv')
         df_rooms = pd.read_csv('datasets/dim_rooms.csv')
         df_agg_bookings = pd.read_csv('datasets/fact_aggregated_bookings.csv')
```

```
In [30]: df_hotels.shape
Out[30]: (25, 4)
```

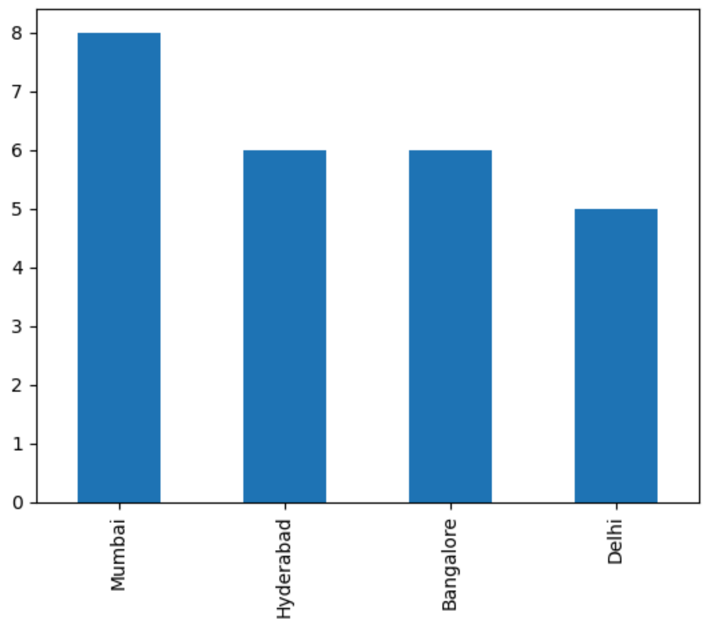
```
In [31]: df_hotels.head(3)
Out[31]:
```

	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 [32]: df_hotels.category.value_counts()
Out[32]: Luxury      16
         Business     9
         Name: category, dtype: int64
```

In [33]:
```
df_hotels.city.value_counts().plot(kind="bar")
```

Out[33]:
```
<Axes: >
```



Exercise: Explore aggregate bookings

In [34]:
```
df_agg_bookings.head(3)
```

Out[34]:

	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 [35]:
```
# write your code here
df_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)
```

Exercise-2. Find out total bookings per property_id

In [36]:
```
# write your code here
df_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
```

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

In [37]:
```
# write your code here
df_agg_bookings[df_agg_bookings['successful_bookings']>df_agg_bookings['capacity']]
```

Out[37]:

	property_id	check_in_date	room_category	successful_bookings	capacity
3	17558	1-May-22	RT1	30	19.0
12	16563	1-May-22	RT1	100	41.0
4136	19558	11-Jun-22	RT2	50	39.0
6209	19560	2-Jul-22	RT1	123	26.0
8522	19559	25-Jul-22	RT1	35	24.0
9194	18563	31-Jul-22	RT4	20	18.0

Exercise-4. Find out properties that have highest capacity

In [38]:
```
# write your code here
(df_agg_bookings['capacity'].max())
```

Out[38]:
```
50.0
```

==> 2. Data Cleaning

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

Out[39]:

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

Out[40]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booki
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	1.0	Ch
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	NaN	
17924	May122218559RT44	18559	12/5/2022	12/5/2022	14-05-22	-10.0	RT4	direct online	NaN	
18020	May122218561RT22	18561	8/5/2022	12/5/2022	14-05-22	-12.0	RT2	makeyourtrip	NaN	
18119	May122218562RT311	18562	5/5/2022	12/5/2022	17-05-22	-6.0	RT3	direct offline	5.0	Ch
18121	May122218562RT313	18562	10/5/2022	12/5/2022	17-05-22	-4.0	RT3	direct online	NaN	
56715	Jun082218562RT12	18562	5/6/2022	8/6/2022	13-06-22	-17.0	RT1	others	NaN	Ch
119765	Jul202219560RT220	19560	19-07-22	20-07-22	22-07-22	-1.0	RT2	others	NaN	Ch
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	-4.0	RT4	logtrip	2.0	Ch

As you can see above, number of guests having less than zero value represents data error. We can ignore these records.

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

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

```
Out[42]: (134578, 12)
```

(2) Outlier removal in revenue generated

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

```
Out[43]: (6500, 28560000)
```

```
In [44]: df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median()
```

```
Out[44]: (15378.036937686695, 13500.0)
```

```
In [45]: avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.std()
```

```
In [46]: higher_limit = avg + 3*std
         higher_limit
```

```
Out[46]: 294498.50173198653
```

```
In [47]: lower_limit = avg - 3*std
         lower_limit
```

```
Out[47]: -263742.4278566132
```

```
In [48]: df_bookings[df_bookings.revenue_generated<=0]
```

Out[48]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	rever

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

Out[49]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booki
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	5.0	Ch
111	May012216559RT32	16559	29-04-22	1/5/2022	2/5/2022	6.0	RT3	direct online	NaN	Ch
315	May012216562RT22	16562	28-04-22	1/5/2022	4/5/2022	2.0	RT2	direct offline	3.0	Ch
562	May012217559RT118	17559	26-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	2.0	RT2	direct online	3.0	Ch

```
In [50]: df_bookings = df_bookings[df_bookings.revenue_generated<=higher_limit]
         df_bookings.shape
```

```
Out[50]: (134573, 12)
```

```
In [51]: df_bookings.revenue_realized.describe()
```

```
Out[51]: 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 [52]:
```
higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std()
higher_limit
```

Out[52]:
```
33479.3586618449
```

In [53]:
```
df_bookings[df_bookings.revenue_realized>higher_limit]
```

Out[53]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	bookir
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	4.0	RT4	others	NaN	Ch
139	May012216559RT43	16559	1/5/2022	1/5/2022	2/5/2022	6.0	RT4	tripster	3.0	Ch
143	May012216559RT47	16559	28-04-22	1/5/2022	3/5/2022	3.0	RT4	others	5.0	Ch
149	May012216559RT413	16559	24-04-22	1/5/2022	7/5/2022	5.0	RT4	logtrip	NaN	Ch
222	May012216560RT45	16560	30-04-22	1/5/2022	3/5/2022	5.0	RT4	others	3.0	Ch
...	...	...	...	...	...	...	...	...	...	
134328	Jul312219560RT49	19560	31-07-22	31-07-22	2/8/2022	6.0	RT4	direct online	5.0	Ch
134331	Jul312219560RT412	19560	31-07-22	31-07-22	1/8/2022	6.0	RT4	others	2.0	Ch
134467	Jul312219562RT45	19562	28-07-22	31-07-22	1/8/2022	6.0	RT4	makeyourtrip	4.0	Ch
134474	Jul312219562RT412	19562	25-07-22	31-07-22	6/8/2022	5.0	RT4	direct offline	5.0	Ch
134581	Jul312217564RT42	17564	31-07-22	31-07-22	1/8/2022	4.0	RT4	makeyourtrip	4.0	Ch

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 [54]:
```
df_bookings[df_bookings.room_category=="RT4"].revenue_realized.describe()
```

Out[54]:
```
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 [55]:
```
# mean + 3*standard deviation
23439+3*9048
```

Out[55]:
```
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 [56]:
```
df_bookings[df_bookings.booking_id=="May012216558RT213"]
```

Out[56]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	rever

In [57]:
```
df_bookings.isnull().sum()
```

Out[57]:
```
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

In [ ]:

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 [58]:
```
# write your code here
df_agg = pd.read_csv('datasets/fact_aggregated_bookings.csv')

df_agg.capacity.fillna(df_agg.capacity.mean(),inplace=True)
df_agg.isnull().sum()
```

Out[58]:
```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               0
dtype: int64
```

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

In [59]:
```
# write your code here
df_agg[df_agg.successful_bookings>df_agg_bookings.capacity]
```

Out[59]:

	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 [60]:
```
df_agg_bookings.head(3)
```

Out[60]:

	property_id	check_in_date	room_category	successful_bookings	capacity
0	16559	1-May-22	RT1	25	30.0
1	19562	1-May-22	RT1	28	30.0
2	19563	1-May-22	RT1	23	30.0

In [61]:
```
df_agg_bookings['occ_pct'] = df_agg_bookings.apply(lambda row: row['successful_bookings']/row['capacity'], axis=1)
```

You can use following approach to get rid of SettingWithCopyWarning

In [62]:
```
new_col = df_agg_bookings.apply(lambda row: row['successful_bookings']/row['capacity'], axis=1)
df_agg_bookings = df_agg_bookings.assign(occ_pct=new_col.values)
df_agg_bookings.head(3)
```

Out[62]:

	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

Convert it to a percentage value

In [63]:
```
df_agg_bookings['occ_pct'] = df_agg_bookings['occ_pct'].apply(lambda x: round(x*100, 2))
df_agg_bookings.head(3)
```

Out[63]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
0	16559	1-May-22	RT1	25	30.0	83.33
1	19562	1-May-22	RT1	28	30.0	93.33
2	19563	1-May-22	RT1	23	30.0	76.67

In [64]:
```
df_bookings.head()
```

Out[64]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancell
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	Checked C
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	Checked C
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	Cancell
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	NaN	No Sh

In [65]:
```
df_agg_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9200 entries, 0 to 9199
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   property_id          9200 non-null   int64  
 1   check_in_date        9200 non-null   object 
 2   room_category        9200 non-null   object 
 3   successful_bookings  9200 non-null   int64  
 4   capacity             9198 non-null   float64
 5   occ_pct              9198 non-null   float64
dtypes: float64(2), int64(2), object(2)
memory usage: 431.4+ 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 [66]:
```
df_agg_bookings.head(3)
```

Out[66]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
0	16559	1-May-22	RT1	25	30.0	83.33
1	19562	1-May-22	RT1	28	30.0	93.33
2	19563	1-May-22	RT1	23	30.0	76.67

In [67]:
```
df_agg_bookings.groupby("room_category")["occ_pct"].mean()
```

Out[67]:
```
room_category
RT1    58.224247
RT2    58.040278
RT3    58.028213
RT4    59.300461
Name: occ_pct, dtype: float64
```

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

In [68]:
```
df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_on="room_id")
df.head(4)
```

Out[68]:

	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	17558	1-May-22	RT1	30	19.0	157.89	RT1	Standard

In [69]:
```
df.drop("room_id",axis=1, inplace=True)
df.head(4)
```

Out[69]:

	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 [70]:
```
df.groupby("room_class")["occ_pct"].mean()
```

Out[70]:
```
room_class
Elite           58.040278
Premium         58.028213
Presidential    59.300461
Standard        58.224247
Name: occ_pct, dtype: float64
```

In [71]:
```
df[df.room_class=="Standard"].occ_pct.mean()
```

Out[71]:
```
58.22424717145344
```

2. Print average occupancy rate per city

In [72]:
```
df_hotels.head(3)
```

Out[72]:

	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 [73]:
```
df = pd.merge(df, df_hotels, on="property_id")
df.head(3)
```

Out[73]:

	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

In [74]:
```
df.groupby("city")["occ_pct"].mean()
```

Out[74]:
```
city
Bangalore    56.594207
Delhi        61.606467
Hyderabad    58.144651
Mumbai       57.936305
Name: occ_pct, dtype: float64
```

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

In [75]:
```
df_date.head(3)
```

Out[75]:

	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 [76]:
```
df = pd.merge(df, df_date, left_on="check_in_date", right_on="date")
df.head(3)
```

Out[76]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	date	mmm yy	week no
0	16559	10-May-22	RT1	18	30.0	60.00	Standard	Atliq Exotica	Luxury	Mumbai	10-May-22	May 22	W 20
1	16559	10-May-22	RT2	25	41.0	60.98	Elite	Atliq Exotica	Luxury	Mumbai	10-May-22	May 22	W 20
2	16559	10-May-22	RT3	20	32.0	62.50	Premium	Atliq Exotica	Luxury	Mumbai	10-May-22	May 22	W 20

In [77]:
```
df.groupby("day_type")["occ_pct"].mean().round(2)
```

Out[77]:
```
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 [78]:
```
df_june_22 = df[df["mmm yy"]=="Jun 22"]
df_june_22.head(4)
```

Out[78]:

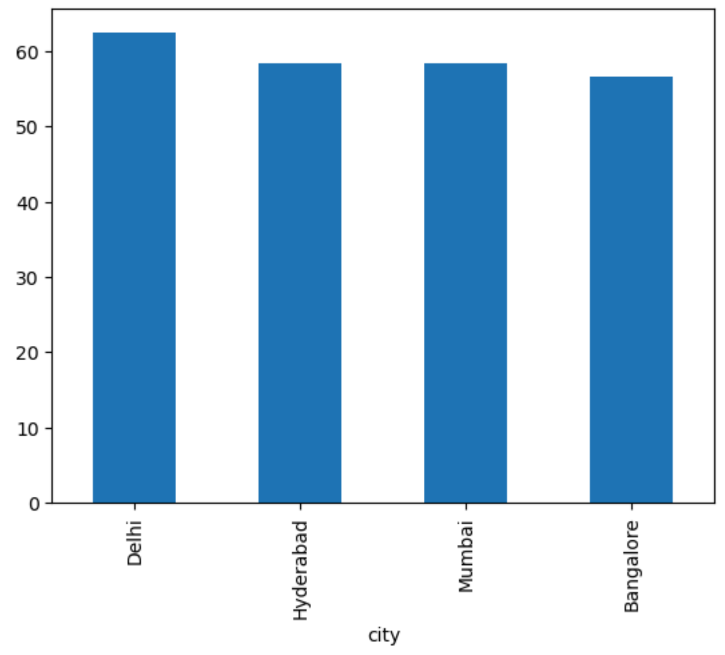
	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	date	mmm yy	we
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury	Mumbai	10-Jun-22	Jun 22	W
2201	16559	10-Jun-22	RT2	26	41.0	63.41	Elite	Atliq Exotica	Luxury	Mumbai	10-Jun-22	Jun 22	W
2202	16559	10-Jun-22	RT3	20	32.0	62.50	Premium	Atliq Exotica	Luxury	Mumbai	10-Jun-22	Jun 22	W
2203	16559	10-Jun-22	RT4	11	18.0	61.11	Presidential	Atliq Exotica	Luxury	Mumbai	10-Jun-22	Jun 22	W

In [79]:
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).sort_values(ascending=False)
```

Out[79]:
```
city
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Bangalore    56.58
Name: occ_pct, dtype: float64
```

In [80]:
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).sort_values(ascending=False).plot(kind="bar")
```

Out[80]:
```
<Axes: xlabel='city'>
```



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

In [81]:
```
df_august = pd.read_csv("datasets/new_data_august.csv")
df_august.head(3)
```

Out[81]:

	property_id	property_name	category	city	room_category	room_class	check_in_date	mmm yy	week no	day_type	successful_bookings	capacity	c
0	16559	Atliq Exotica	Luxury	Mumbai	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	30	30	1(
1	19562	Atliq Bay	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	21	30	1
2	19563	Atliq Palace	Business	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	23	30	1

In [82]:
```
df_august.columns
```
Out[82]:
```
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 [83]:
```
df.columns
```
Out[83]:
```
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 [84]:
```
df_august.shape
```
Out[84]:
```
(7, 13)
```

In [85]:
```
df.shape
```
Out[85]:
```
(6500, 14)
```

In [86]:
```
latest_df = pd.concat([df, df_august], ignore_index = True, axis = 0)
latest_df.tail(10)
```
Out[86]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	date	mmm yy
6497	18560	31-Jul-22	RT2	34	40.0	85.00	Elite	Atliq City	Business	Hyderabad	31-Jul-22	Jul 22
6498	18560	31-Jul-22	RT3	17	24.0	70.83	Premium	Atliq City	Business	Hyderabad	31-Jul-22	Jul 22
6499	18560	31-Jul-22	RT4	12	15.0	80.00	Presidential	Atliq City	Business	Hyderabad	31-Jul-22	Jul 22
6500	16559	01-Aug-22	RT1	30	30.0	NaN	Standard	Atliq Exotica	Luxury	Mumbai	NaN	Aug-22
6501	19562	01-Aug-22	RT1	21	30.0	NaN	Standard	Atliq Bay	Luxury	Bangalore	NaN	Aug-22
6502	19563	01-Aug-22	RT1	23	30.0	NaN	Standard	Atliq Palace	Business	Bangalore	NaN	Aug-22
6503	19558	01-Aug-22	RT1	30	40.0	NaN	Standard	Atliq Grands	Luxury	Bangalore	NaN	Aug-22
6504	19560	01-Aug-22	RT1	20	26.0	NaN	Standard	Atliq City	Business	Bangalore	NaN	Aug-22
6505	17561	01-Aug-22	RT1	18	26.0	NaN	Standard	Atliq Blu	Luxury	Mumbai	NaN	Aug-22
6506	17564	01-Aug-22	RT1	10	16.0	NaN	Standard	Atliq Seasons	Business	Mumbai	NaN	Aug-22

In [87]:
```
latest_df.shape
```
Out[87]:
```
(6507, 15)
```

6. Print revenue realized per city

In [88]:
```
df_bookings.head()
```
Out[88]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancell
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	Checked C
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	Checked C
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	Cancell
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	NaN	No Sh

In [89]:
```
df_hotels.head(3)
```
Out[89]:

	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 [90]:
```
df_bookings_all = pd.merge(df_bookings, df_hotels, on="property_id")
df_bookings_all.head(3)
```

Out[90]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancell
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	Checked C
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	Checked C

In [91]:
```
df_bookings_all.groupby("city")["revenue_realized"].sum()
```

Out[91]:
```
city
Bangalore    420383550
Delhi        294404488
Hyderabad    325179310
Mumbai       668569251
Name: revenue_realized, dtype: int64
```

7. Print month by month revenue

In [92]:
```
df_date.head(3)
```

Out[92]:

	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 [93]:
```
df_date["mmm yy"].unique()
```

Out[93]:
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

In [94]:
```
df_bookings_all.head(3)
```

Out[94]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancell
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	Checked C
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	Checked C

In [95]:
```
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 [96]:
```
df_date["date"] = pd.to_datetime(df_date["date"])
df_date.head(3)
```

Out[96]:

	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 [97]:
```
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 [98]:
```
df_bookings_all["check_in_date"] = pd.to_datetime(df_bookings_all["check_in_date"])
df_bookings_all.head(4)
```

Out[98]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat
0	May012216558RT12	16558	30-04-22	2022-01-05	2/5/2022	2.0	RT1	others	NaN	Cancell
1	May012216558RT15	16558	27-04-22	2022-01-05	2/5/2022	4.0	RT1	direct online	5.0	Checked C
2	May012216558RT16	16558	1/5/2022	2022-01-05	3/5/2022	2.0	RT1	others	4.0	Checked C
3	May012216558RT17	16558	28-04-22	2022-01-05	6/5/2022	2.0	RT1	others	NaN	Cancell

In [99]:
```
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on="check_in_date", right_on="date")
df_bookings_all.head(3)
```

Out[99]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat
0	May052216558RT11	16558	15-04-22	2022-05-05	7/5/2022	3.0	RT1	tripster	5.0	Checked C
1	May052216558RT12	16558	30-04-22	2022-05-05	7/5/2022	2.0	RT1	others	NaN	Cancell
2	May052216558RT13	16558	1/5/2022	2022-05-05	6/5/2022	3.0	RT1	direct offline	5.0	Checked C

In [100]:
```
df_bookings_all.groupby("mmm yy")["revenue_realized"].sum()
```

Out[100]:
```
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 [101]:
```
# write your code here
df_bookings_all.groupby(df_bookings_all.property_name)['revenue_realized'].sum().round(2).sort_values(ascending=False)
```

Out[101]:
```
property_name
Atliq Exotica    219076161
Atliq Palace     209474575
Atliq City       196555383
Atliq Bay        179416721
Atliq Blu        179203544
Atliq Grands     145860641
Atliq Seasons     45920757
Name: revenue_realized, dtype: int64
```

Exercise-2 Print average rating per city

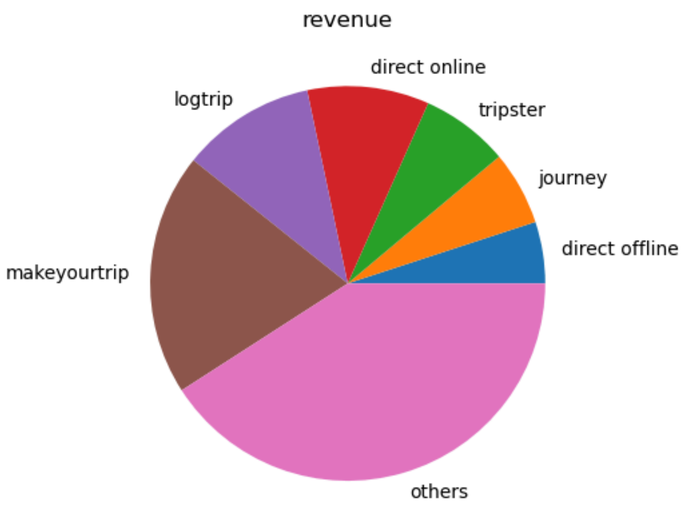
In [102]:
```
# write your code here
df_bookings_all.groupby(df_bookings_all.city)['ratings_given'].mean().round(1).sort_values()
```

Out[102]:
```
city
Bangalore    3.4
Mumbai       3.6
Hyderabad    3.7
Delhi        3.8
Name: ratings_given, dtype: float64
```

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

In [106]:
```
code here
all.groupby(df_bookings_all.booking_platform)['revenue_realized'].sum().sort_values().plot(kind="pie",title="revenue",ylabel=
```

Out[106]:
```
<Axes: title={'center': 'revenue'}, ylabel=' '>
```


revenue
direct online
logtrip
tripster
journey
direct offline
makeyourtrip
others