AtliQ Hotels Data Analysis Project

Atliq Hotels, a luxury hotel brand in India with locations in Mumbai, Hyderabad, Delhi, and Bangalore, is seeing a downturn in business. To solve this issue, they have offered a dataset encompassing three months from May 2022 to July 2022 for examination, as well as supplementary data for August 2022. This notebook seeks to analyze data and provide insights based on the findings.

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

Improting Necessary Libraries

In [7]:
```
import pandas as pd
import numpy as np
```

---

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 dataframe.

In [8]:
```
df_bookings = pd.read_csv('datasets/fact_bookings.csv')
```

Explore bookings data using the Head() function.

In [247…
```
df_bookings.head()
```

Out[247…

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

Identifying the total number of rows and columns using Shape() function.

df_bookings.shape

Lists the unique room category present in bookings data using Unique() function.

In [20]:
```
df_bookings.room_category.unique()
```

Out[20]:
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

Lists of distinct booking platforms found in bookings data using Unique() fuction.

In [250…
```
df_bookings.booking_platform.unique()
```

Out[250…
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

Count of booking per platform using Value_count() function.

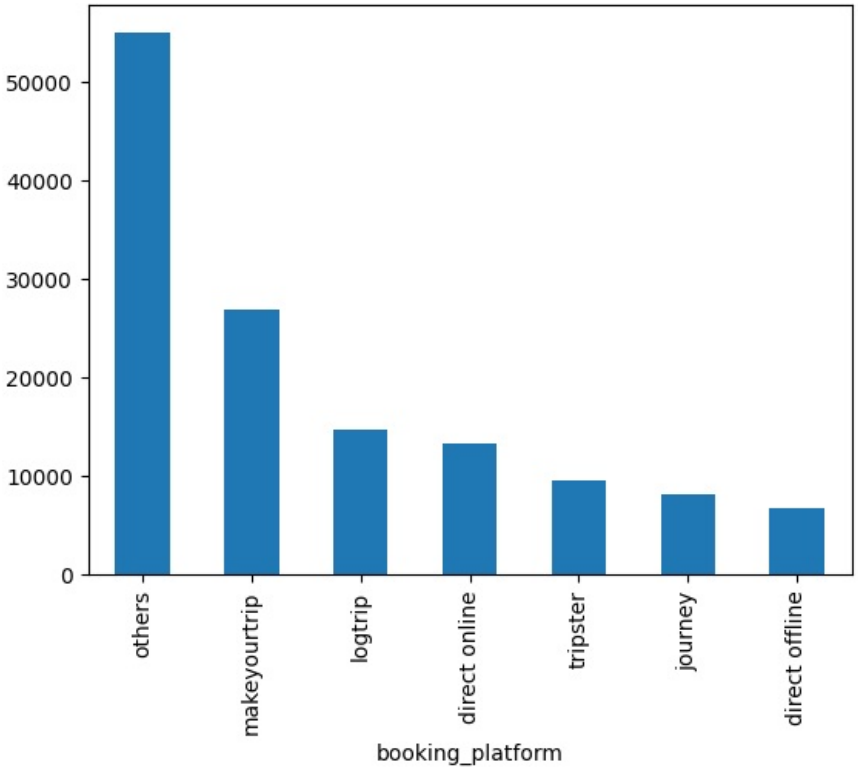
In [22]:
```
df_bookings.booking_platform.value_counts()
```

Out[22]:
```
booking_platform
others            55066
makeyourtrip      26898
logtrip           14756
direct online     13379
tripster           9630
journey            8106
direct offline     6755
Name: count, dtype: int64
```

Creates a bar chart showing the distribution of booking platform in the dataset usint Plot() function.

In [24]:
```
df_bookings.booking_platform.value_counts().plot(kind="bar")
```

Out[24]:
```
<Axes: xlabel='booking_platform'>
```



Privides a summary of descriptive statistics for the bookings dataframe using Describe() function.

In [253…
```
df_bookings.describe()
```

Out[253…

	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 [9]:
```
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')
```

Provides the number of rows and columns in the hotels dataframe using Shape() function.

In [255…
```
df_hotels.shape
```

Out[255…
```
(25, 4)
```

Displays the first few rows of the hotels dtaframe using Head() function.

In [7]:
```
df_hotels.head(3)
```

Out[7]:

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

Provide a count of each property_category in the hotels dataframe using the Value_count() fuction.

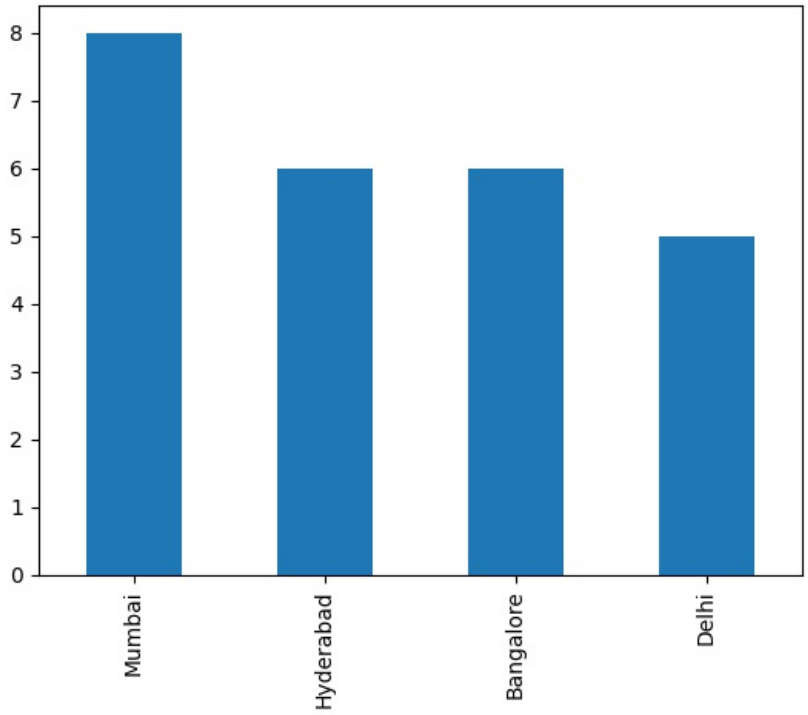
In [257…
```
df_hotels.category.value_counts()
```

Out[257…
```
Luxury      16
Business     9
Name: category, dtype: int64
```

Plot the number of hotels per city with a bar chart using the Plot() function.

In [258…
```
df_hotels.city.value_counts().plot(kind="bar")
```

Out[258…
```
<AxesSubplot: >
```



Explore aggregate bookings data

In [6]:
```
df_agg_bookings.head(3)
```

Out[6]:

	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

Find out unique property ids in aggregate bookings dataset

In [7]:
```
df_agg_bookings.property_id.unique()
```

Out[7]:
```
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])
```

Exercise-2. Find out total bookings per property_id

In [14]:
```
df_agg_bookings.groupby("property_id")['successful_bookings'].sum()
```

Out[14]:
```
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
```

Find out days on which bookings are greater than capacity

In [18]:
```
df_agg_bookings[df_agg_bookings.successful_bookings > df_agg_bookings.capacity ]
```

Out[18]:

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

Find out properties that have highest capacity

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

Out[20]:

	property_id	check_in_date	room_category	successful_bookings	capacity
27	17558	1-May-22	RT2	38	50.0
128	17558	2-May-22	RT2	27	50.0
229	17558	3-May-22	RT2	26	50.0
328	17558	4-May-22	RT2	27	50.0
428	17558	5-May-22	RT2	29	50.0
...	...	...	...	...	...
8728	17558	27-Jul-22	RT2	22	50.0
8828	17558	28-Jul-22	RT2	21	50.0
8928	17558	29-Jul-22	RT2	23	50.0
9028	17558	30-Jul-22	RT2	32	50.0
9128	17558	31-Jul-22	RT2	30	50.0

92 rows × 5 columns

---

2. Data Cleaning

Provide a summary of descriptive statistic for bookings dataframe using the Describe() function.

In [265…
```
df_bookings.describe()
```

Out[265…

	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

Filters bookings where the number of guests is less than or equal to zero.

In [266…
```
df_bookings[df_bookings.no_guests<=0]
```

Out[266…

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

Filters the bookings dataframe to include entries with more than zero guests.

In [267…
```
df_bookings = df_bookings[df_bookings.no_guests>0]
```

In [268…
```
df_bookings.shape
```

Out[268… (134578, 12)

(2) Outlier removal in revenue generated

Calculates the minimum and maximum revenue generated in the bookings dataframe

In [269…
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

Out[269… (6500, 28560000)

Calculates the mean and median revenue generated in the bookings dataframe

In [270…
```
df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median()
```

Out[270… (15378.036937686695, 13500.0)

Calculates the mean and standard deviation of revenue generated in the bookings dataframe.

In [271…
```
avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.std()
```

Calculate the upper limit using the formula: Higher_limit = avg + 3*std.

In [272…
```
higher_limit = avg + 3*std
higher_limit
```

Out[272… 294498.50173207896

Calculate the lower limit using the formula: Lower_limit = avg - 3*std.

In [273…
```
lower_limit = avg - 3*std
lower_limit
```

Out[273… -263742.4278567056

We have no values in the revenue realised column that are less than or equal to zero.

In [274…
```
df_bookings[df_bookings.revenue_generated<=0]
```

Out[274…

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_give

Filters bookings where revenue generated exceeds a specified higher limit.

In [275…
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

Out[275…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platfor
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtr
111	May012216559RT32	16559	29-04-22	1/5/2022	2/5/2022	6.0	RT3	direct onli
315	May012216562RT22	16562	28-04-22	1/5/2022	4/5/2022	2.0	RT2	direct offli
562	May012217559RT118	17559	26-04-22	1/5/2022	2/5/2022	2.0	RT1	othe
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	2.0	RT2	direct onli

Filters the bookings dataframe to include only rows where revenue generated is less than or equal to a specified higher limit, and then displays the shape of the filterd dataframe.

In [276…
```
df_bookings = df_bookings[df_bookings.revenue_generated<=higher_limit]
df_bookings.shape
```

Out[276… (134573, 12)

Removing outliers in revenue_realized.

Generate summary statistics for the revenue_realized in the bookings dataframe.

In [277…
```
df_bookings.revenue_realized.describe()
```

Out[277…
```
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 [278…
```
higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std()
higher_limit
```

Out[278… 33479.358661845814

Displays booking where revenue realized exceeds the higher limit.

In [279…
```
df_bookings[df_bookings.revenue_realized>higher_limit]
```

Out[279…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platfor
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	4.0	RT4	othe
139	May012216559RT43	16559	1/5/2022	1/5/2022	2/5/2022	6.0	RT4	tripst
143	May012216559RT47	16559	28-04-22	1/5/2022	3/5/2022	3.0	RT4	othe
149	May012216559RT413	16559	24-04-22	1/5/2022	7/5/2022	5.0	RT4	logtr
222	May012216560RT45	16560	30-04-22	1/5/2022	3/5/2022	5.0	RT4	othe
...	...	...	...	...	...	...	...	
134328	Jul312219560RT49	19560	31-07-22	31-07-22	2/8/2022	6.0	RT4	direct onlir
134331	Jul312219560RT412	19560	31-07-22	31-07-22	1/8/2022	6.0	RT4	othe
134467	Jul312219562RT45	19562	28-07-22	31-07-22	1/8/2022	6.0	RT4	makeyourtr
134474	Jul312219562RT412	19562	25-07-22	31-07-22	6/8/2022	5.0	RT4	direct offlir
134581	Jul312217564RT42	17564	31-07-22	31-07-22	1/8/2022	4.0	RT4	makeyourtr

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

Describes the statistcal summary of revenue_realised for room category "RT4" in the bookings dataframe.

In [280…
```
df_bookings[df_bookings.room_category=="RT4"].revenue_realized.describe()
```

Out[280…
```
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 [281…
```
# mean + 3*standard deviation
23439+3*9048
```

Out[281… 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 [282…
```
df_bookings[df_bookings.booking_id=="May012216558RT213"]
```

Out[282…

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_give

Shows the count of missing values in each column of the bookings dataframe.

In [283…
```
df_bookings.isnull().sum()
```

Out[283…
```
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 [12]:
```
df_agg_bookings = pd.read_csv("datasets/fact_aggregated_bookings.csv")
```

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)

Counts the number of missing values in the aggregated bookings dataframe.

In [13]:
```
df_agg_bookings.isnull().sum()
```

Out[13]:
```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               2
dtype: int64
```

Filters the df_agg_bookings dataframe to show where the 'capacity' column is NaN.

In [60]:
```
df_agg_bookings[df_agg_bookings.capacity.isna()]
```

Out[60]:

	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

Replaces null values in the 'capacity' column of df_agg_bookings using the median value.

In [11]:
```
df_agg_bookings.capacity.fillna(df_agg_bookings.capacity.median(), inplace = True)
df_agg_bookings.head(15)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_19192\2200157660.py:1: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df_agg_bookings.capacity.fillna(df_agg_bookings.capacity.median(), inplace = True)
```

Out[11]:

	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
5	17560	1-May-22	RT1	28	40.0
6	19558	1-May-22	RT1	25	40.0
7	19560	1-May-22	RT1	23	26.0
8	17561	1-May-22	RT1	22	25.0
9	16560	1-May-22	RT1	24	34.0
10	16561	1-May-22	RT1	16	18.0
11	16562	1-May-22	RT1	20	31.0
12	16563	1-May-22	RT1	100	41.0
13	17559	1-May-22	RT1	26	32.0
14	17562	1-May-22	RT1	12	25.0

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

Filters aggregated bookings where the number of successful bookings exceeds the capacity.

In [14]:
```
df_agg_bookings[df_agg_bookings.successful_bookings > df_agg_bookings.capacity]
```

Out[14]:

	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

Filters df_agg_bookings to include only rows where the number of successful bookings is less then or equal to the capacity.

In [16]:
```
df_agg_bookings = df_agg_bookings[df_agg_bookings.successful_bookings <= df_agg_bookings.capacity]
df_agg_bookings.shape
```

Out[16]: (9080, 5)

---

3. Data Transformation

In [106…
```
df_agg_bookings.head(3)
```

Out[106…

	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 [107…
```
df_agg_bookings['occ_pct'] = df_agg_bookings.apply(lambda row: row['successful_bookings']/row['capacity'], axis
```

Create a new column to indicate the occupancy_percentage.

In [108…
```
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[108…

	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 [109…
```
df_agg_bookings['occ_pct'] = df_agg_bookings['occ_pct'].apply(lambda x: round(x*100, 2))
df_agg_bookings.head(3)
```

Out[109…

	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 [111…
```
df_agg_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9082 entries, 0 to 9199
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   property_id          9082 non-null   int64  
 1   check_in_date        9082 non-null   object 
 2   room_category        9082 non-null   object 
 3   successful_bookings  9082 non-null   int64  
 4   capacity             9082 non-null   float64
 5   occ_pct              9082 non-null   float64
dtypes: float64(2), int64(2), object(2)
memory usage: 496.7+ KB
```

In [18]:
```
df_agg_bookings.check_in_date = pd.to_datetime(df_agg_bookings['check_in_date'],format = '%d-%b-%y')
```

In [19]:
```
df_agg_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9080 entries, 0 to 9199
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   property_id          9080 non-null   int64
 1   check_in_date        9080 non-null   datetime64[ns]
 2   room_category        9080 non-null   object
 3   successful_bookings  9080 non-null   int64
 4   capacity             9080 non-null   float64
dtypes: datetime64[ns](1), float64(1), int64(2), object(1)
memory usage: 425.6+ KB
```

---

4. Insights Generation

---

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

In [112…
```
df_agg_bookings.head(3)
```

Out[112…

	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

Calcualte the average occupancy percentage for each room category.

In [113…
```
df_agg_bookings.groupby("room_category")["occ_pct"].mean()
```

Out[113…
```
room_category
RT1    57.779310
RT2    57.752486
RT3    57.604256
RT4    58.017915
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

Joins the aggregated bookings data with room details, then previews the first few rows of the combined dataframe.

In [114…
```
df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_on="room_id")
df.head(4)
```

Out[114…

	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

Deletes the room_id coumn from the dataframe df.

In [115…
```
df.drop("roaom_id",axis=1, inplace=True)
df.head(4)
```

Out[115…

	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

Calcualtes the average occupancy percentage for each room_class.

In [116…
```
df.groupby("room_class")["occ_pct"].mean()
```

Out[116…
```
room_class
Elite           57.752486
Premium         57.604256
Presidential    58.017915
Standard        57.779310
Name: occ_pct, dtype: float64
```

In [117…
```
df[df.room_class=="Standard"].occ_pct.mean()
```

Out[117…
```
np.float64(57.77931004366812)
```

2. Print average occupancy rate per city

In [118…
```
df_hotels.head(3)
```

Out[118…

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

Joins df and df_hotels on property_id and displays the first few rows.

In [119…
```
df = pd.merge(df, df_hotels, on="property_id")
df.head(3)
```

Out[119…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	
0	16559	1-May-22	RT1	25	30.0	83.33	Standard	Atliq Exotica	Luxury	M
1	19562	1-May-22	RT1	28	30.0	93.33	Standard	Atliq Bay	Luxury	Ban
2	19563	1-May-22	RT1	23	30.0	76.67	Standard	Atliq Palace	Business	Ban

Calcualtes the average occupancy percentage for each city.

In [120…
```
df.groupby("city")["occ_pct"].mean()
```

Out[120…
```
city
Bangalore    56.033283
Delhi        60.629588
Hyderabad    57.795562
Mumbai       57.343912
Name: occ_pct, dtype: float64
```

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

In [87]:
```
df_date.head(3)
```

Out[87]:

	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

Joins df and df_date on check_in_date and displays the first few rows.

In [121…
```
df = pd.merge(df, df_date, left_on="check_in_date", right_on="date")
df.head(3)
```

Out[121…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	
0	19563	10-May-22	RT3	15	29.0	51.72	Premium	Atliq Palace	Business	Bar
1	18560	10-May-22	RT1	19	30.0	63.33	Standard	Atliq City	Business	Hyd
2	19562	10-May-22	RT1	18	30.0	60.00	Standard	Atliq Bay	Luxury	Bar

Calculates the mean occuupancy percentage by day_type, rounded to two decimal places.

In [122…
```
df.groupby("day_type")["occ_pct"].mean().round(2)
```

Out[122…
```
day_type
weekeday    50.86
weekend     71.33
Name: occ_pct, dtype: float64
```

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

In [123…
```
df_june_22 = df[df["mmm yy"]=="Jun 22"]
df_june_22.head(4)
```

Out[123…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category
2177	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury
2178	19562	10-Jun-22	RT1	19	30.0	63.33	Standard	Atliq Bay	Luxury
2179	19563	10-Jun-22	RT1	17	30.0	56.67	Standard	Atliq Palace	Business
2180	17558	10-Jun-22	RT1	9	19.0	47.37	Standard	Atliq Grands	Luxury

Calulates and sorts the average occupancy percentages by city in descending order.

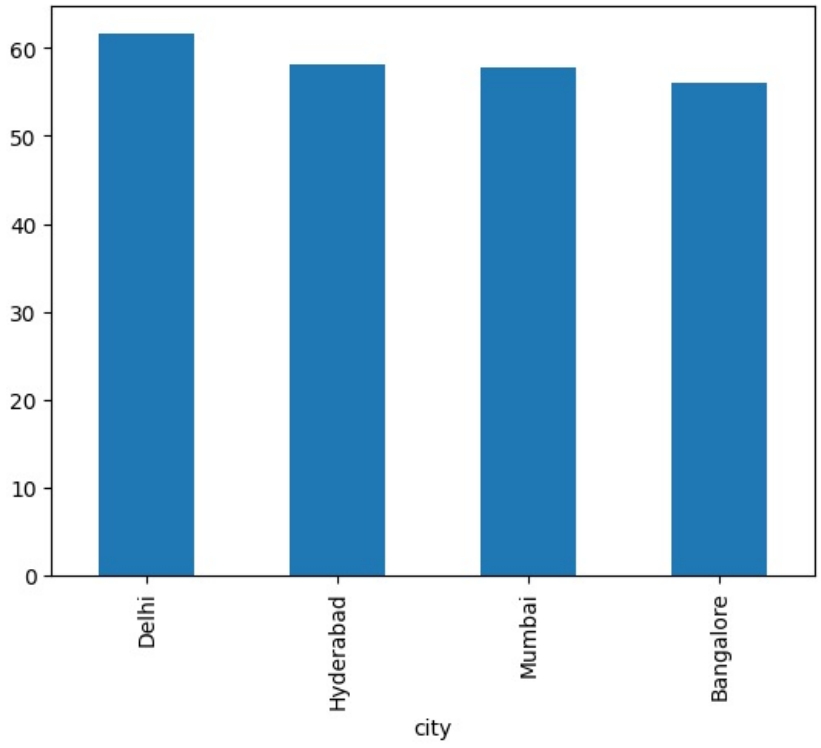
In [91]:
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).sort_values(ascending=False)
```

Out[91]:
```
city
Delhi        61.65
Hyderabad    58.21
Mumbai       57.82
Bangalore    56.00
Name: occ_pct, dtype: float64
```

Generates a bar plot of the average occupancy percentage by city, sorted in descending order, using June 2022 data.

In [124…
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).sort_values(ascending=False).plot(kind="bar")
```

Out[124…
```
<Axes: xlabel='city'>
```



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

In [125…
```
df_august = pd.read_csv("datasets/new_data_august.csv")
df_august.head(3)
```

Out[125…

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

In [95]:
```
df_august.columns
```

Out[95]:
```
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 [96]:
```
df.columns
```

Out[96]:
```
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 [126…
```
df_august.shape
```

Out[126…
```
(7, 13)
```

In [98]:
```
df.shape
```

Out[98]:
```
(6428, 14)
```

Concatenates 'df' and 'df_august' into latest_df, resetting the index to maintain continuity

In [127…
```
latest_df = pd.concat([df, df_august], ignore_index = True, axis = 0)
latest_df.tail(10)
```

Out[127…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category
6425	17558	31-Jul-22	RT4	3	6.0	50.0	Presidential	Atliq Grands	Luxury
6426	19563	31-Jul-22	RT4	3	6.0	50.0	Presidential	Atliq Palace	Business
6427	17561	31-Jul-22	RT4	3	4.0	75.0	Presidential	Atliq Blu	Luxury
6428	16559	01-Aug-22	RT1	30	30.0	NaN	Standard	Atliq Exotica	Luxury
6429	19562	01-Aug-22	RT1	21	30.0	NaN	Standard	Atliq Bay	Luxury
6430	19563	01-Aug-22	RT1	23	30.0	NaN	Standard	Atliq Palace	Business
6431	19558	01-Aug-22	RT1	30	40.0	NaN	Standard	Atliq Grands	Luxury
6432	19560	01-Aug-22	RT1	20	26.0	NaN	Standard	Atliq City	Business
6433	17561	01-Aug-22	RT1	18	26.0	NaN	Standard	Atliq Blu	Luxury
6434	17564	01-Aug-22	RT1	10	16.0	NaN	Standard	Atliq Seasons	Business

In [128…
```
latest_df.shape
```

Out[128…
```
(6435, 15)
```

6. Print revenue realized per city

In [129…
```
df_bookings.head()
```

Out[129…

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

In [345…
```
df_hotels.head(3)
```

Out[345…

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

Merges the df_bookings and df_hotels dataframes on property_id to create df_bookings_all.

In [142…
```
df_bookings_all = pd.merge(df_bookings, df_hotels, on="property_id")
df_bookings_all.head(3)
```

Out[142…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	rati
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	

In [143…
```
df_bookings_all.groupby("city")["revenue_realized"].sum()
```

Out[143…
```
city
Bangalore    420397050
Delhi        294500318
Hyderabad    325232870
Mumbai       668640991
Name: revenue_realized, dtype: int64
```

7. Print month by month revenue

In [144…
```
df_date.head(3)
```

Out[144…

	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 [145…
```
df_date["mmm yy"].unique()
```

Out[145…
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

In [146…
```
df_bookings_all.head(3)
```

Out[146…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	rati
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	

In [147…
```
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 [148…
```
df_date["date"] = pd.to_datetime(df_date["date"])
df_date.head(3)
```

Out[148…

	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 [149…
```
df_bookings_all.info()
```

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

Converts the check_in_date coumn in df_booking_all to datetime format, handling errors by coercing invalid dates.

In [152…
```
df_bookings_all["check_in_date"] = pd.to_datetime(df_bookings_all["check_in_date"], dayfirst=True, errors='coer

df_bookings_all.head(4)
```

Out[152…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	rati
0	May012216558RT11	16558	27-04-22	2022-05-01	2/5/2022	-3.0	RT1	direct online	
1	May012216558RT12	16558	30-04-22	2022-05-01	2/5/2022	2.0	RT1	others	
2	May012216558RT13	16558	28-04-22	2022-05-01	4/5/2022	2.0	RT1	logtrip	
3	May012216558RT14	16558	28-04-22	2022-05-01	2/5/2022	-2.0	RT1	others	

Joins the bookings dataframe df_bookings_all with the dataframe df_date using the check_in_date and date coumns.

In [153…
```
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on="check_in_date", right_on="date")
df_bookings_all.head(3)
```

Out[153…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	rati
0	May012216558RT11	16558	27-04-22	2022-05-01	2/5/2022	-3.0	RT1	direct online	
1	May012216558RT12	16558	30-04-22	2022-05-01	2/5/2022	2.0	RT1	others	
2	May012216558RT13	16558	28-04-22	2022-05-01	4/5/2022	2.0	RT1	logtrip	

Calculates the aggregate revenue_realized for each month.

In [154…
```
df_bookings_all.groupby("mmm yy")["revenue_realized"].sum()
```

Out[154…
```
mmm yy
Jul 22    243180932
Jun 22    229644140
May 22    234516453
Name: revenue_realized, dtype: int64
```

Print revenue realized per hotel type

In [158…
```
df_hotels.head()
```

Out[158…

	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 [160…
```
df_bookings_all= pd.merge(df_bookings_all, df_hotels, on ='property_id')
df_bookings_all.head()
```

Out[160…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	rati
0	May012216558RT11	16558	27-04-22	2022-05-01	2/5/2022	-3.0	RT1	direct online	
1	May012216558RT12	16558	30-04-22	2022-05-01	2/5/2022	2.0	RT1	others	
2	May012216558RT13	16558	28-04-22	2022-05-01	4/5/2022	2.0	RT1	logtrip	
3	May012216558RT14	16558	28-04-22	2022-05-01	2/5/2022	-2.0	RT1	others	
4	May012216558RT15	16558	27-04-22	2022-05-01	2/5/2022	4.0	RT1	direct online	

5 rows × 25 columns

In [161…
```
df_bookings_all.groupby('property_name_y')['revenue_realized'].sum()
```

Out[161…
```
property_name_y
Atliq Bay        107557972
Atliq Blu        108111729
Atliq City       118290783
Atliq Exotica    133673106
Atliq Grands      87316569
Atliq Palace     125553143
Atliq Seasons     26838223
Name: revenue_realized, dtype: int64
```

Print average rating per city

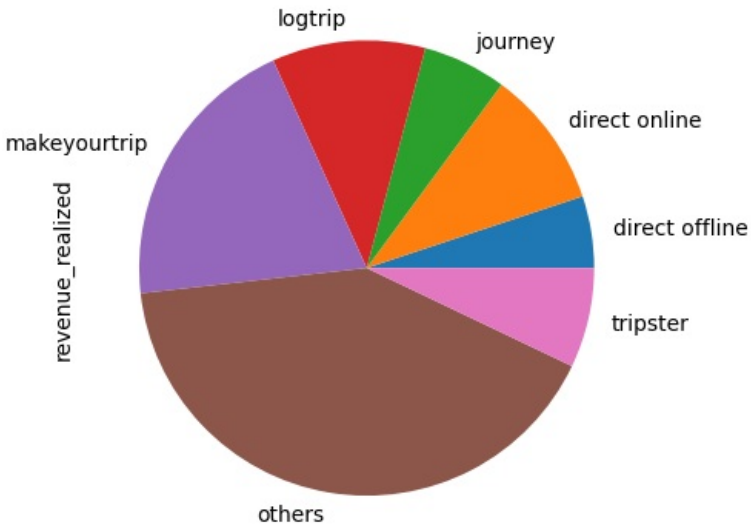
In [162…
```
df_bookings_all.groupby('city')['ratings_given'].mean()
```

Out[162…
```
city
Bangalore    3.414599
Delhi        3.787587
Hyderabad    3.654123
Mumbai       3.655835
Name: ratings_given, dtype: float64
```

Print a pie chart of revenue realized per booking platform

In [165…
```
df_bookings_all.groupby('booking_platform')['revenue_realized'].sum().plot(kind = "pie")
```

Out[165…
```
<Axes: ylabel='revenue_realized'>
```


logtrip
journey
direct online
makeyourtrip
direct offline
revenue_realized
tripster
others

In [ ]: