
HOSPITALITY
DATA ANALYSIS
PRESENTED BY
AVULA DILIP KUMAR

AtliQ Hotels Data Analysis Project

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

==> 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

Load the bookings data in a dataframe

[2]:
```
df_bookings = pd.read_csv('datasets/fact_bookings.csv')
```

Explore bookings data

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

[3] :

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

	revenue_generated	revenue_realized
0	10010	10010
1	9100	3640
2	9100000	9100
3	9100	3640
4	10920	10920

Show number of rows and columns in the data

[4] :
```
df_bookings.shape
```

[4]: (134590, 12)

Show different room categories

[5] :
```
df_bookings.room_category.unique()
```

[5] : array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)

Show different booking platforms

[6] :
```
df_bookings.booking_platform.unique()
```

[6] :
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

Show booking count for each platform

[7] :
```
df_bookings.booking_platform.value_counts()
```

[7] :
```
booking_platform
others            55066
makeyourtrip      26898
logtrip           14756
direct  online    13379
tripster           9630
journey            8106
direct  offline    6755
Name: count, dtype: int64
```

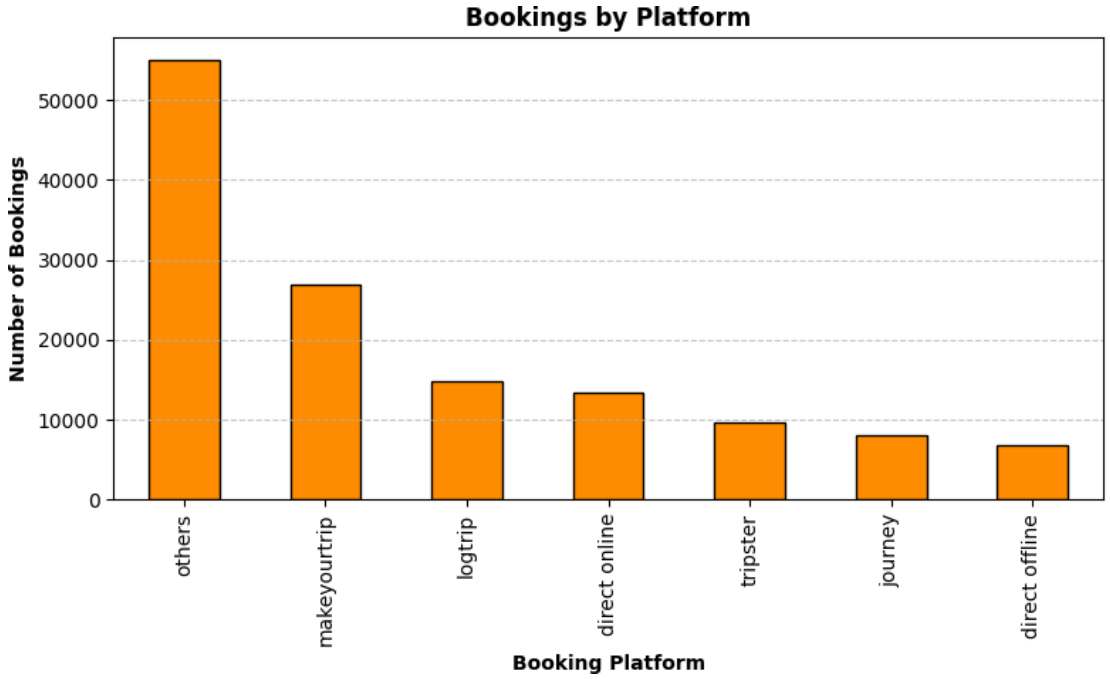
Plot bookings by platform

[8] :
```
platform_counts = df_bookings.booking_platform.value_counts()
```

```
plt.figure(figsize=(8,5))
platform_counts.plot(kind="bar", color="darkorange", edgecolor="black")

plt.title("Bookings by Platform", fontsize=12, fontweight="bold", color="black")
plt.xlabel("Booking Platform", fontsize=10, fontweight="bold", color="black")
plt.ylabel("Number of Bookings", fontsize=10, fontweight="bold", color="black")

plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.tight_layout()
plt.show()
```



Display statistical summary of bookings dataset

[9] :
```
df_bookings.describe()
```

[9] :

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

	revenue_realized
count	134590.000000
mean	12696.123256
std	6928.108124
min	2600.000000
25%	7600.000000
50%	11700.000000
75%	15300.000000
max	45220.000000

Load additional datasets (date, hotels, rooms, aggregated bookings)

[10] :
```
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')
```

Show number of rows and columns in hotels dataset

[11] :
```
df_hotels.shape
```

[11]: (25, 4)

Display first 3 rows of hotels dataset

[12] :
```
df_hotels.head(3)
```

[12]:

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

Count of hotels by category

[13] :
```
df_hotels.category.value_counts()
```

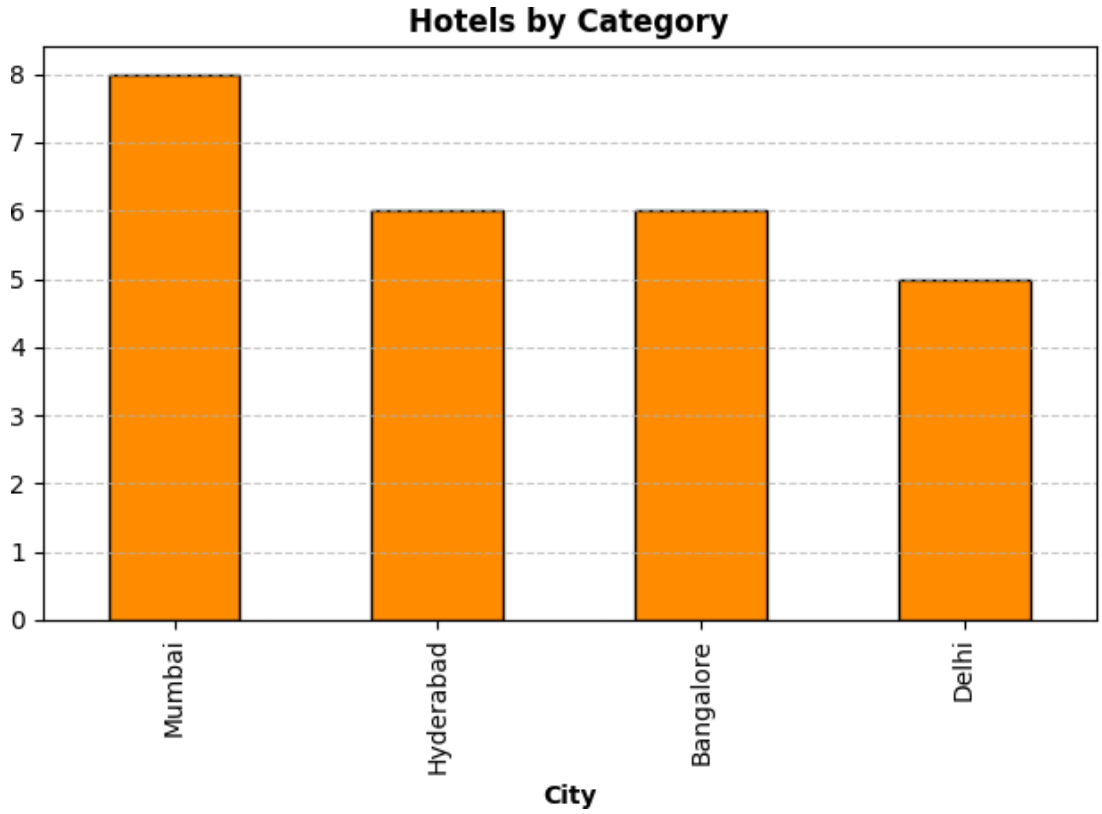
[13] :
```
category
Luxury      16
Business     9
Name: count, dtype: int64
```

[14] :
```
df_hotels.city.value_counts().plot(kind="bar", color="darkorange",␣
 ↪edgecolor="black")

plt.title("Hotels by Category", fontsize=12, fontweight="bold", color="black")
plt.xlabel("City", fontsize=10, fontweight="bold", color="black")

plt.grid(axis="y", linestyle="--", alpha=0.7)
```

```
plt.tight_layout()
plt.show()
```



Display first 3 rows of agg_bookings dataset

[15] :
```
df_agg_bookings.head(3)
```

[15] :

	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

1. ***Unique Property IDs in the Aggregated Bookings Dataset***

[16] :
```
df_agg_bookings.property_id.unique()
```

[16] :
```
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])
```

2. Total Bookings Count per Property ID

[17] :
```
df_agg_bookings.groupby('property_id')['successful_bookings'].sum()
```

[17]:
```
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
```

3. Days When Bookings Exceeded Capacity

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

[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

4. Properties with the Highest Capacity

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

[19] :
```
np.float64(50.0)
```

==> 2. Data Cleaning

Summary Statistics of Bookings Dataset

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

[20]:

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

	revenue_realized
count	134590.000000
mean	12696.123256
std	6928.108124
min	2600.000000
25%	7600.000000
50%	11700.000000
75%	15300.000000
max	45220.000000

Bookings with Zero or Negative Guests

[21] :
```
df_bookings[df_bookings.no_guests<=0]
```

[21]:

	booking_id	property_id	booking_date	check_in_date
0	May012216558RT11	16558	27-04-22	1/5/2022
3	May012216558RT14	16558	28-04-22	1/5/2022
17924	May122218559RT44	18559	12/5/2022	12/5/2022
18020	May122218561RT22	18561	8/5/2022	12/5/2022
18119	May122218562RT311	18562	5/5/2022	12/5/2022
18121	May122218562RT313	18562	10/5/2022	12/5/2022
56715	Jun082218562RT12	18562	5/6/2022	8/6/2022
119765	Jul202219560RT220	19560	19-07-22	20-07-22
134586	Jul312217564RT47	17564	30-07-22	31-07-22

	checkout_date	no_guests	room_category	booking_platform	ratings_given
0	2/5/2022	-3.0	RT1	direct online	1.0
3	2/5/2022	-2.0	RT1	others	NaN
17924	14-05-22	-10.0	RT4	direct online	NaN

18020	14-05-22	-12.0	RT2	makeyourtrip	NaN
18119	17-05-22	-6.0	RT3	direct offline	5.0
18121	17-05-22	-4.0	RT3	direct online	NaN
56715	13-06-22	-17.0	RT1	others	NaN
119765	22-07-22	-1.0	RT2	others	NaN
134586	1/8/2022	-4.0	RT4	logtrip	2.0

	booking_status	revenue_generated	revenue_realized
0	Checked Out	10010	10010
3	Cancelled	9100	3640
17924	No Show	20900	20900
18020	Cancelled	9000	3600
18119	Checked Out	16800	16800
18121	Cancelled	14400	5760
56715	Checked Out	6500	6500
119765	Checked Out	13500	13500
134586	Checked Out	38760	38760

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

Remove Invalid Guest Entries

[22] :
```
df_bookings = df_bookings[df_bookings.no_guests>0]
```

Dataset Shape After Cleaning

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

[23]: (134578, 12)

Minimum and Maximum Revenue Generated

[24] :
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

[24] : (np.int64(6500), np.int64(28560000))

Mean and Median Revenue Generated

[25] :
```
df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median()
```

[25] : (np.float64(15378.036937686695), np.float64(13500.0))

Mean and Standard Deviation of Revenue

[26] :
```
avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.
  std()
```

Calculate Higher Limit for Revenue Generated

[27] :
```
higher_limit = avg + 3*std
higher_limit
```

[27] : np.float64(294498.50173207896)

Calculate Lower Limit for Revenue Generated

[28] :
```
lower_limit = avg - 3*std
lower_limit
```

[28] : np.float64(-263742.4278567056)

Identify Records Above Higher Limit for Revenue Generated

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

[29]:

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

	checkout_date	no_guests	room_category	booking_platform	ratings_given
2	4/5/2022	2.0	RT1	logtrip	5.0
111	2/5/2022	6.0	RT3	direct online	NaN
315	4/5/2022	2.0	RT2	direct offline	3.0
562	2/5/2022	2.0	RT1	others	NaN
129176	29-07-22	2.0	RT2	direct online	3.0

	booking_status	revenue_generated	revenue_realized
2	Checked Out	9100000	9100
111	Checked Out	28560000	28560
315	Checked Out	12600000	12600
562	Cancelled	2000000	4420
129176	Checked Out	10000000	12600

Remove Outliers Based on Higher Limit for Revenue Generated

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

[30]: (134573, 12)

Revenue Realized Summary Statistics

[31] :
```
df_bookings.revenue_realized.describe()
```

[31]:
```
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
```

Calculate Higher Limit for Revenue Realized

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

[32] : np.float64(33479.358661845814)

Identify Records Above Higher Limit for Revenue Realized

[33] :
```
df_bookings[df_bookings.revenue_realized>higher_limit]
```

[33]:

	booking_id	property_id	booking_date	check_in_date
137	May012216559RT41	16559	27-04-22	1/5/2022
139	May012216559RT43	16559	1/5/2022	1/5/2022
143	May012216559RT47	16559	28-04-22	1/5/2022
149	May012216559RT413	16559	24-04-22	1/5/2022
222	May012216560RT45	16560	30-04-22	1/5/2022
...	...	...	...	...
134328	Jul312219560RT49	19560	31-07-22	31-07-22
134331	Jul312219560RT412	19560	31-07-22	31-07-22
134467	Jul312219562RT45	19562	28-07-22	31-07-22
134474	Jul312219562RT412	19562	25-07-22	31-07-22
134581	Jul312217564RT42	17564	31-07-22	31-07-22

	checkout_date	no_guests	room_category	booking_platform	ratings_given
137	7/5/2022	4.0	RT4	others	NaN
139	2/5/2022	6.0	RT4	tripster	3.0
143	3/5/2022	3.0	RT4	others	5.0
149	7/5/2022	5.0	RT4	logtrip	NaN
222	3/5/2022	5.0	RT4	others	3.0
...	...	...	...	...	...
134328	2/8/2022	6.0	RT4	direct online	5.0
134331	1/8/2022	6.0	RT4	others	2.0
134467	1/8/2022	6.0	RT4	makeyourtrip	4.0
134474	6/8/2022	5.0	RT4	direct offline	5.0
134581	1/8/2022	4.0	RT4	makeyourtrip	4.0

	booking_status	revenue_generated	revenue_realized
137	Checked Out	38760	38760
139	Checked Out	45220	45220
143	Checked Out	35530	35530
149	Checked Out	41990	41990
222	Checked Out	34580	34580
...	...	...	...
134328	Checked Out	39900	39900
134331	Checked Out	39900	39900
134467	Checked Out	39900	39900
134474	Checked Out	37050	37050
134581	Checked Out	38760	38760

[1299 rows x 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

Revenue Statistics for RT4 Room Category

[34] :
```
df_bookings[df_bookings.room_category=="RT4"].revenue_realized.describe()
```

[34]:
```
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
```

[35] :
```
# mean + 3*standard deviation
23439+3*9048
```

[35]:
```
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

Missing Values in df_bookings

[36] :
```
df_bookings.isnull().sum()
```

[37]:
```
booking_id            0
property_id           0
booking_date          0
check_in_date         0
checkout_date         0
no_guests             0
room_category         0
booking_platform      0
ratings_given     77897
booking_status        0
revenue_generated     0
revenue_realized      0
dtype: int64
```

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

Missing Values in df_agg_bookings

[37] :
```
df_agg_bookings.isnull().sum()
```

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

Rows with Missing Capacity in df_agg_bookings

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

[39] :

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

Median Capacity in df_agg_bookings

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

[40] :
```
np.float64(25.0)
```

Fill Missing Capacity with Median & Display Specific Rows

[41]:
```
df_agg_bookings.fillna(df_agg_bookings.capacity.median(),inplace=True)
df_agg_bookings.loc[[8,15]]
```

[41]:

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

Where Successful Bookings Exceed Capacity

[42]:
```
df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.capacity]
```

[42]:

	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

First 3 Rows of df_agg_bookings

[43]:
```
df_agg_bookings.head(3)
```

[43]:

	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

Create Occupancy Percentage Column (Method 1)

[44]:
```
df_agg_bookings['occ_pct'] = df_agg_bookings.apply(lambda row:
 row['successful_bookings']/row['capacity'], axis=1)
```

Create Occupancy Percentage Column (Method 2 - assign)

[45]:
```
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)
```

[45]:

	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 to Percentage & Round

[46]:

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

[46]:

	property_id	check_in_date	room_categ	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

[47]:

```
df_bookings.head()
```

[47]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_generated	revenue_realized
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancelled	9100	3640
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	Checked Out	10920	10920
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	Checked Out	9100	9100
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	Cancelled		
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	NaN	No Show		

6	9100	3640
7	9100	9100

Dataset Info After Adding Column

[48] :
```
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            9200 non-null   float64
 5   occ_pct             9200 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.Mean Occupancy Percentage by Room Category

[49] :
```
df_agg_bookings.groupby("room_category")["occ_pct"].mean()
```

[49] :
```
room_category
RT1    58.232748
RT2    58.040278
RT3    58.028213
RT4    59.300461
Name: occ_pct, dtype: float64
```

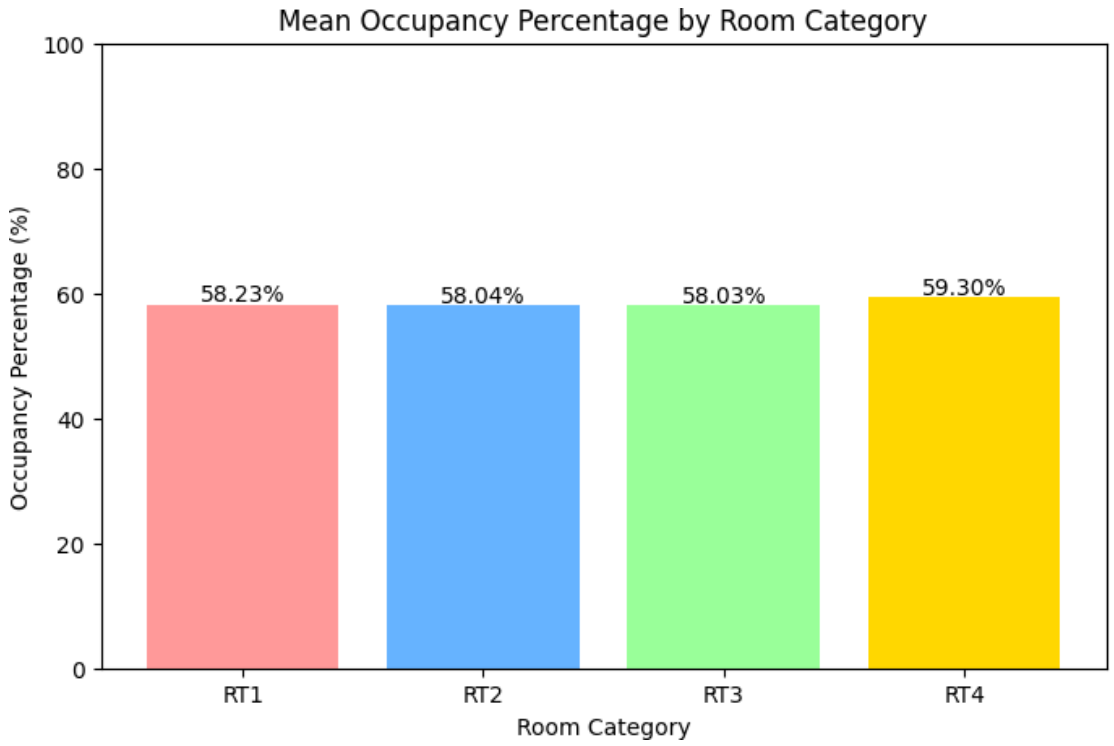
[50] :
```
room_categories = ['RT1', 'RT2', 'RT3', 'RT4']
occ_pct = [58.232748, 58.040278, 58.028213, 59.300461]

colors = ['#FF9999', '#66B3FF', '#99FF99', '#FFD700']
```

```
plt.figure(figsize=(8,5))
bars = plt.bar(room_categories, occ_pct, color=colors)
plt.title('Mean Occupancy Percentage by Room Category')
plt.xlabel('Room Category')
plt.ylabel('Occupancy Percentage (%)')
plt.ylim(0, 100)

for bar, value in zip(bars, occ_pct):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.5,
             f'{value:.2f}%', ha='center', fontsize=10)

plt.show()
```



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

Merge Bookings with Room Data

[51] :
```
df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category",␣
  ↪right_on="room_id")
df.head(4)
```

[51] :

	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

Drop 'room_id' Column

[52] :

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

[52]:

	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

Mean Occupancy Percentage by Room Class

[53] :

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

[53] :

```
room_class
Elite           58.040278
Premium         58.028213
Presidential    59.300461
Standard        58.232748
Name: occ_pct, dtype: float64
```

Mean Occupancy Percentage for Standard Room Class

[54] :

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

[54] :

```
np.float64(58.23274782608696)
```

2.Print Average Occupancy Rate per City

[55] :

```
df_hotels.head(3)
```

[55]:

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

Merge Data with Hotels Data

[56] :
```
df = pd.merge(df, df_hotels, on="property_id")
df.head(3)
```

[56]:

	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

	occ_pct	room_class	property_name	category	city
0	83.33	Standard	Atliq Exotica	Luxury	Mumbai
1	93.33	Standard	Atliq Bay	Luxury	Bangalore
2	76.67	Standard	Atliq Palace	Business	Bangalore

Mean Occupancy Percentage by City

[57]:
```
df.groupby("city")["occ_pct"].mean()
```

[57]:
```
city
Bangalore    56.594207
Delhi        61.606467
Hyderabad    58.144651
Mumbai       57.943142
Name: occ_pct, dtype: float64
```

3.When Was the Occupancy Better? Weekday or Weekend?

[58]:
```
df_date.head(3)
```

[58]:

	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

Merge DataFrames on Date Columns

[59] :
```
df = pd.merge(df, df_date, left_on="check_in_date", right_on="date")
df.head(3)
```

[59] :

	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
0	19563	10-May-22	RT3	15	29.0	51.72	Premium	Atliq Palace	Business	Bangalore	10-May-22	May 22	W 20	weekeday
1	18560	10-May-22	RT1	19	30.0	63.33	Standard	Atliq City	Business	Hyderabad	10-May-22	May 22	W 20	weekeday
2	19562	10-May-22	RT1	18	30.0	60.00	Standard	Atliq Bay	Luxury	Bangalore	10-May-22	May 22	W 20	weekeday

4.Group by Day Type and Calculate Average Occupancy Percentage

[60] :
```
df.groupby("day_type")["occ_pct"].mean().round(2)
```

[60]:
```
day_type
weekeday    50.90
weekend     72.39
Name: occ_pct, dtype: float64
```

In the Month of June, What Is the Occupancy for Different Cities

[61]:
```
df_june_22 = df[df["mmm yy"]=="Jun 22"]
df_june_22.head(4)
```

[61]:

	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
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury	Mumbai	10-Jun-22	Jun 22	W 24	weekeday
2201	19562	10-Jun-22	RT1	19	30.0	63.33	Standard	Atliq Bay	Luxury	Bangalore	10-Jun-22	Jun 22	W 24	weekeday
2202	19563	10-Jun-22	RT1	17	30.0	56.67	Standard	Atliq Palace	Business	Bangalore	10-Jun-22	Jun 22	W 24	weekeday
2203	17558	10-Jun-22	RT1	9	19.0	47.37	Standard	Atliq Grands	Luxury	Mumbai	10-Jun-22	Jun 22	W 24	weekeday

Average Occupancy Percentage by City (June 2022)

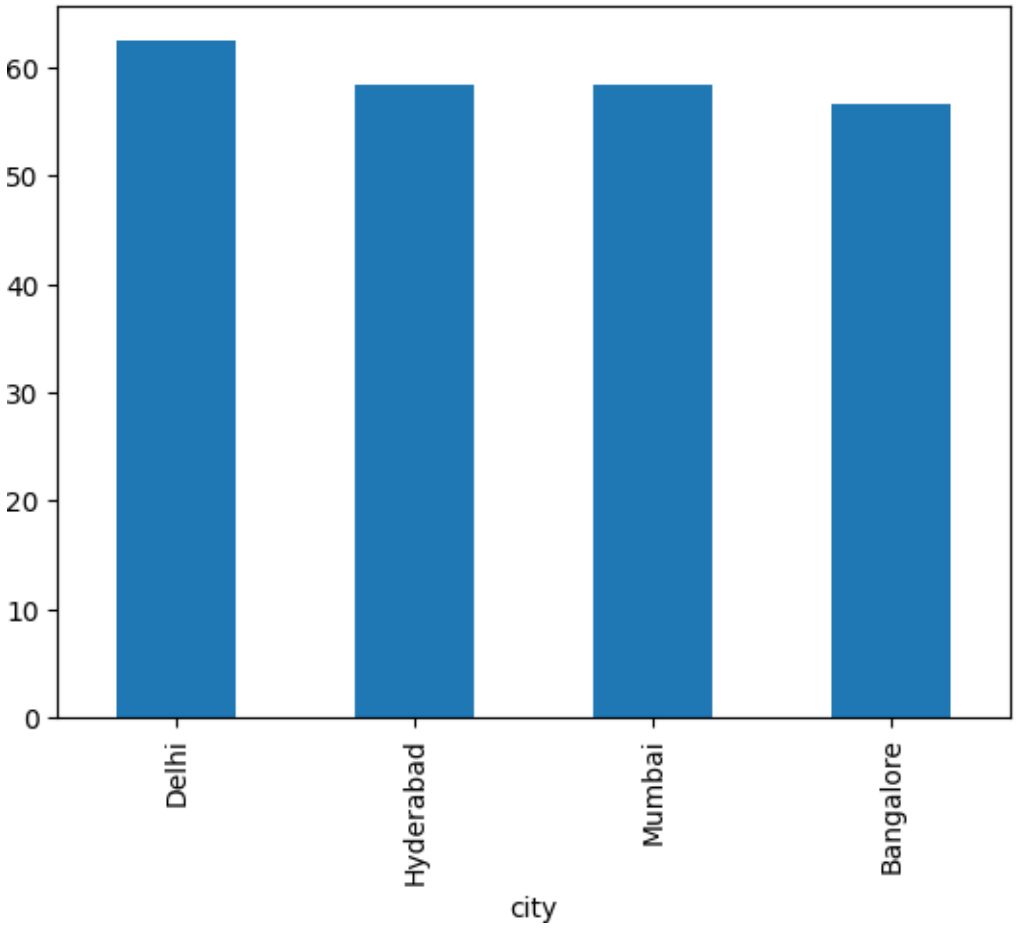
[62]:
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).
  sort_values(ascending=False)
```

[62]:
```
city
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Bangalore    56.58
Name: occ_pct, dtype: float64
```

Bar Plot of Average Occupancy Percentage by City (June 2022)

[63]:
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).
  sort_values(ascending=False).plot(kind="bar")
```

[63] : <Axes: xlabel='city'>



5. Append August Data to the Existing Dataset

[64] :
```
df_august = pd.read_csv("datasets/new_data_august.csv")
df_august.head(3)
```

[64]:

	property_id	property_name	category	city	room_category	room_class	check_in_date	mmm yy	week no	day_type	successful_bookings	capacity	occ%
0	16559	Atliq Exotica	Luxury	Mumbai	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	30	30	100.00
1	19562	Atliq Bay	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	21	30	70.00
2	19563	Atliq Palace	Business	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	23	30	76.67

View DataFrame Column Names

[65] :
```
df_august.columns
```

[65] :
```
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')
```

[66] :
```
df.columns
```

[66] :
```
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')
```

[67] :
```
df_august.shape
```

[67]: (7, 13)

[68] :
```
df.shape
```

[68]: (6500, 14)

Concatenate DataFrames and View Last 10 Records

[69] :
```
latest_df = pd.concat([df, df_august], ignore_index = True, axis = 0)
latest_df.tail(5)
```

[69] :

	property_id	check_in_date	room_category	successful_bookings	capacity
6502	19563	01-Aug-22	RT1	23	30.0
6503	19558	01-Aug-22	RT1	30	40.0
6504	19560	01-Aug-22	RT1	20	26.0
6505	17561	01-Aug-22	RT1	18	26.0
6506	17564	01-Aug-22	RT1	10	16.0

	occ_pct	room_class	property_name	category	city	date	mmm yy
6502	NaN	Standard	Atliq Palace	Business	Bangalore	NaN	Aug-22
6503	NaN	Standard	Atliq Grands	Luxury	Bangalore	NaN	Aug-22
6504	NaN	Standard	Atliq City	Business	Bangalore	NaN	Aug-22
6505	NaN	Standard	Atliq Blu	Luxury	Mumbai	NaN	Aug-22
6506	NaN	Standard	Atliq Seasons	Business	Mumbai	NaN	Aug-22

	week no	day_type	occ%
6502	W 32	weekeday	76.67
6503	W 32	weekeday	75.00
6504	W 32	weekeday	76.92
6505	W 32	weekeday	69.23
6506	W 32	weekeday	62.50

Check Shape of Latest DataFrame

[70] :
```
latest_df.shape
```

[70]: (6507, 15)

6. Print Revenue Realized per City

[71] :
```
df_bookings.head()
```

[71] :

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

	no_guests	room_category	booking_platform	ratings_given	booking_status
1	2.0	RT1	others	NaN	Cancelled
4	4.0	RT1	direct online	5.0	Checked Out
5	2.0	RT1	others	4.0	Checked Out
6	2.0	RT1	others	NaN	Cancelled
7	2.0	RT1	logtrip	NaN	No Show

	revenue_generated	revenue_realized
1	9100	3640
4	10920	10920

5	9100	9100
6	9100	3640
7	9100	9100

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

[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

7.Merge Bookings and Hotels Data

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

[73] :

	booking_id	property_id	booking_date	check_in_date	checkout_date
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022

	no_guests	room_category	booking_platform	ratings_given	booking_status
0	2.0	RT1	others	NaN	Cancelled
1	4.0	RT1	direct online	5.0	Checked Out
2	2.0	RT1	others	4.0	Checked Out

	revenue_generated	revenue_realized	property_name	category	city
0	9100	3640	Atliq Grands	Luxury	Delhi
1	10920	10920	Atliq Grands	Luxury	Delhi
2	9100	9100	Atliq Grands	Luxury	Delhi

8.Total Revenue Realized by City

[74] :
```
df_bookings_all.groupby("city")["revenue_realized"].sum()
```

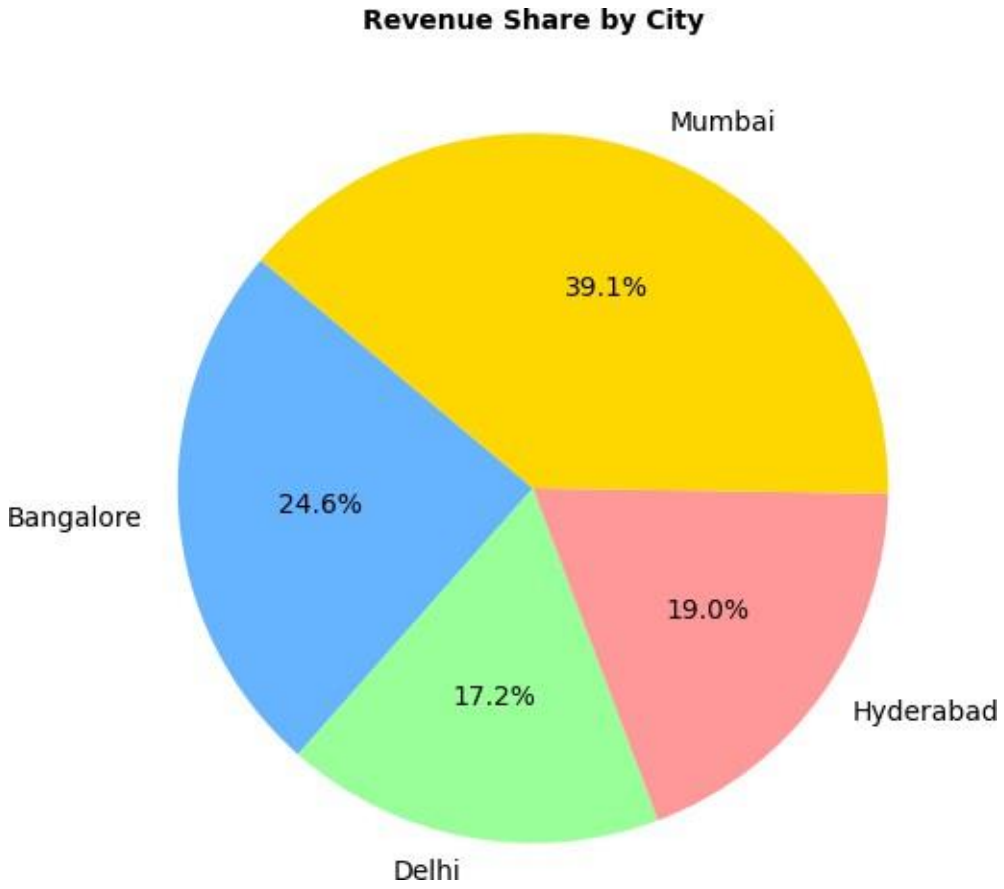
[74] :
```
city
Bangalore    420383550
Delhi        294404488
Hyderabad    325179310
Mumbai       668569251
Name: revenue_realized, dtype: int64
```

Revenue Percentage by Major Cities

[75] :
```
cities = ['Bangalore', 'Delhi', 'Hyderabad', 'Mumbai']
revenue = [420383550, 294404488, 325179310, 668569251]

colors = ['#66B3FF', '#99FF99', '#FF9999', '#FFD700']
```

```
plt.figure(figsize=(6,6))
plt.pie(revenue, labels=cities, autopct='%1.1f%%', startangle=140,
  colors=colors)
plt.title('Revenue Share by City', fontsize=10, fontweight='bold')
plt.show()
```



9.Print Month-by-Month Revenue

[76] :
```
df_date.head(3)
```

[76] :

	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

Unique Month-Year Values in df_date

[77] :
```
df_date["mmm  yy"].unique()
```

[77] :
```
array(['May  22', 'Jun  22', 'Jul  22'], dtype=object)
```

[78] :
```
df_bookings_all.head(3)
```

[78]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	\
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	

	no_guests	room_category	booking_platform	ratings_given	booking_status	\
0	2.0	RT1	others	NaN	Cancelled	
1	4.0	RT1	direct online	5.0	Checked Out	
2	2.0	RT1	others	4.0	Checked Out	

	revenue_generated	revenue_realized	property_name	category	city
0	9100	3640	Atliq Grands	Luxury	Delhi
1	10920	10920	Atliq Grands	Luxury	Delhi
2	9100	9100	Atliq Grands	Luxury	Delhi

DataFrame Info: df_date

[79] :
```
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
```

Convert 'date' Column to Datetime in df_date

[80] :
```
df_date["date"] = pd.to_datetime(df_date["date"],format = "%d-%b-%y")
df_date.head(3)
```

[80] :

	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

DataFrame Info: df_bookings_all

[81] :
```
df_bookings_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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-nullobject 
dtypes: float64(2), int64(3), object(10)
memory usage: 15.4+ MB
```

Convert 'check_in_date' Column to Datetime in df_bookings_all

[82] :
```
df_bookings_all["check_in_date"] =pd.
  sto_datetime(df_bookings_all['check_in_date'],errors='coerce')
df_bookings_all.head(4)
```

[82] :

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

	no_guests	room_category	booking_platform	ratings_given	booking_status	\
0	2.0	RT1	others	NaN	Cancelled	
1	4.0	RT1	direct online	5.0	Checked Out	
2	2.0	RT1	others	4.0	Checked Out	
3	2.0	RT1	others	NaN	Cancelled	

	revenue_generated	revenue_realized	property_name	category	city
0	9100	3640	Atliq Grands	Luxury	Delhi
1	10920	10920	Atliq Grands	Luxury	Delhi
2	9100	9100	Atliq Grands	Luxury	Delhi
3	9100	3640	Atliq Grands	Luxury	Delhi

Merge df_bookings_all with df_date on Dates

[83] :
```
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on="check_in_date",
 right_on="date")
df_bookings_all.head(3)
```

[83]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_generated	revenue_realized	property_name	category	city	date	mmm yy	week no	day_type
0	May052216558RT11	16558	15-04-22	2022-05-05	7/5/2022	3.0	RT1	tripster	5.0	Checked Out	10010	10010	Atliq Grands	Luxury	Delhi	2022-05-05	May 22	W 19	weekeday
1	May052216558RT12	16558	30-04-22	2022-05-05	7/5/2022	2.0	RT1	others	NaN	Cancelled	9100	3640	Atliq Grands	Luxury	Delhi	2022-05-05	May 22	W 19	weekeday
2	May052216558RT13	16558	1/5/2022	2022-05-05	6/5/2022	3.0	RT1	direct offline	5.0	Checked Out	10010	10010	Atliq Grands	Luxury	Delhi	2022-05-05	May 22	W 19	weekeday

Group Revenue by Month-Year

[84] :
```
df_bookings_all.groupby("mmm  yy")["revenue_realized"].sum()
```

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

1. Print revenue realized per hotel type

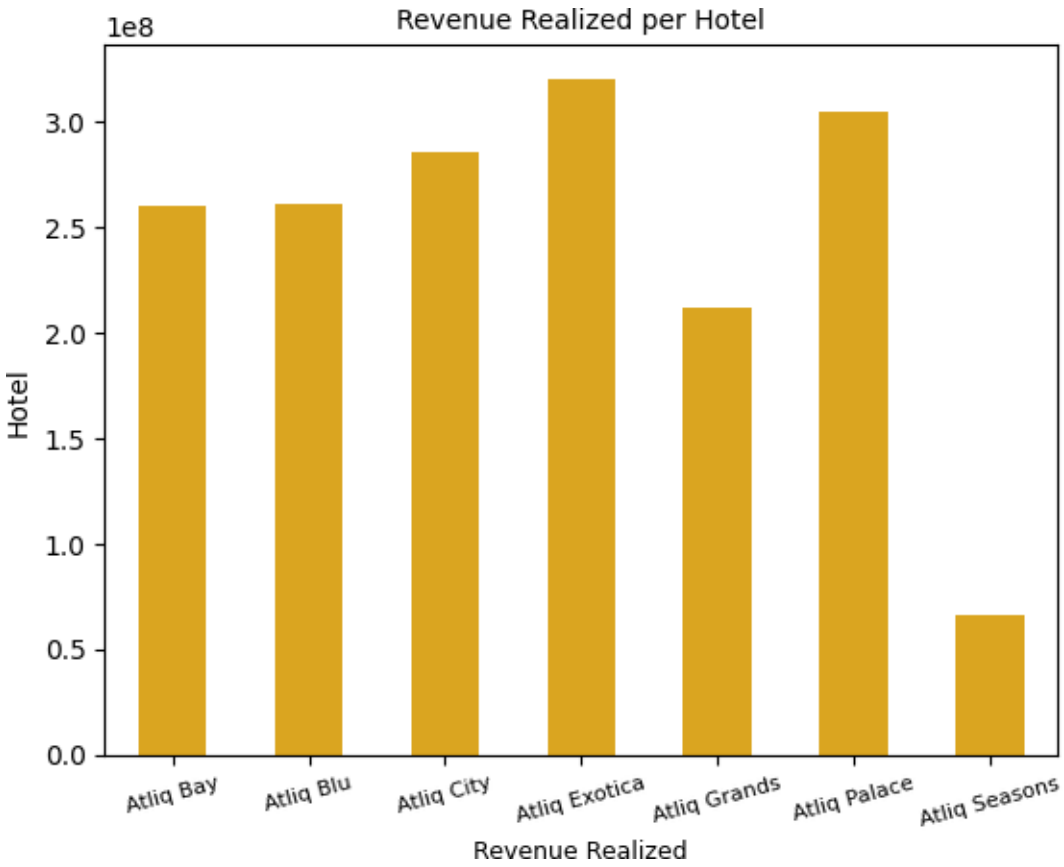
[85] :
```
df = pd.merge(df_bookings,df_hotels,on = "property_id")
df.groupby("property_name")["revenue_realized"].sum()
```

[85] : property_name
```
Atliq  Bay       259996918
Atliq  Blu       260851922
Atliq  City      285798439
Atliq  Exotica   320258588
Atliq  Grands    211462134
Atliq  Palace    304081863
```

```
Atliq  Seasons      66086735
Name: revenue_realized, dtype: int64
```

[86] :
```
df.groupby("property_name")["revenue_realized"].sum().plot(
    kind='bar',
    color='goldenrod')

plt.xlabel("Revenue Realized",fontsize=9)
plt.ylabel("Hotel")
plt.title("Revenue Realized per Hotel",fontsize=10)
plt.xticks(rotation=15,fontsize=8)
plt.show()
```



[87] :
```
df_hotels
df_bookings.head()
```

[87] :

	booking_id	property_id	booking_date	check_in_date	checkout_date
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022

5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022

	no_guests	room_category	booking_platform	ratings_given	booking_status
1	2.0	RT1	others	NaN	Cancelled
4	4.0	RT1	direct online	5.0	Checked Out
5	2.0	RT1	others	4.0	Checked Out
6	2.0	RT1	others	NaN	Cancelled
7	2.0	RT1	logtrip	NaN	No Show

	revenue_generated	revenue_realized
1	9100	3640
4	10920	10920
5	9100	9100
6	9100	3640
7	9100	9100

2. Print average rating per city

[88] :
```
df.groupby("city")["ratings_given"].mean()
```

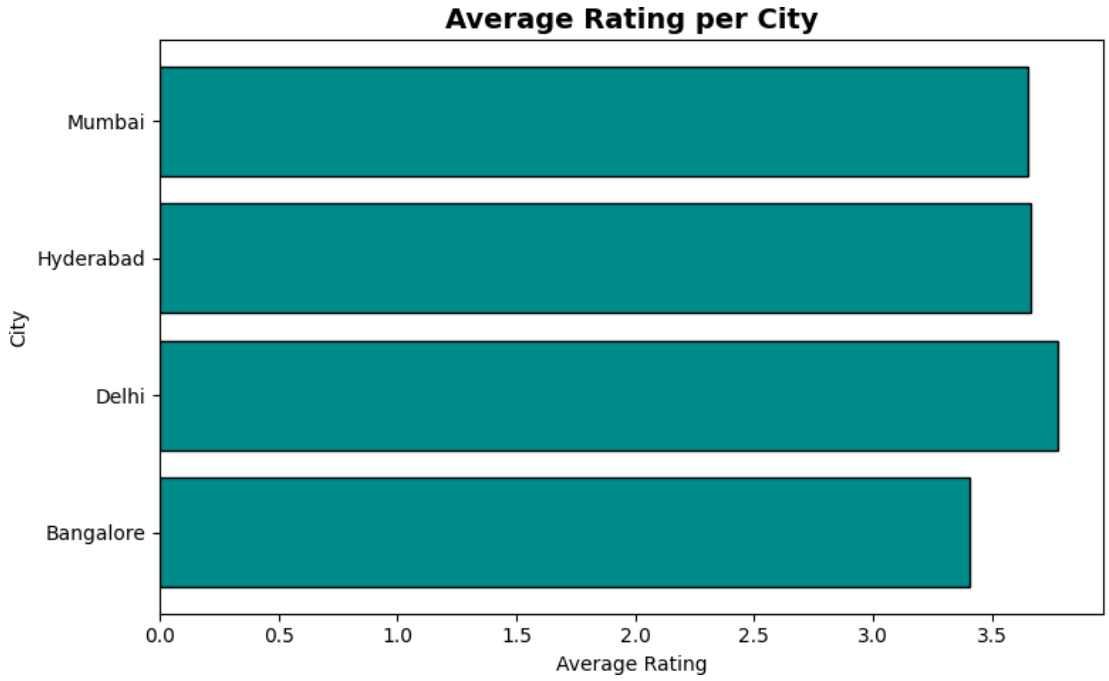
[88] :
```
city
Bangalore    3.407681
Delhi        3.779298
Hyderabad    3.661041
Mumbai       3.650545
Name: ratings_given, dtype: float64
```

[89] :
```
ratings = {
    'Bangalore': 3.407681,
    'Delhi': 3.779298,
    'Hyderabad': 3.661041,
    'Mumbai': 3.650545
}

cities = list(ratings.keys())
avg_ratings = list(ratings.values())

plt.figure(figsize=(8, 5))
plt.barh(cities, avg_ratings, color='darkcyan', edgecolor='black')

plt.xlabel("Average Rating")
plt.ylabel("City")
plt.title("Average Rating per City", fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()
```

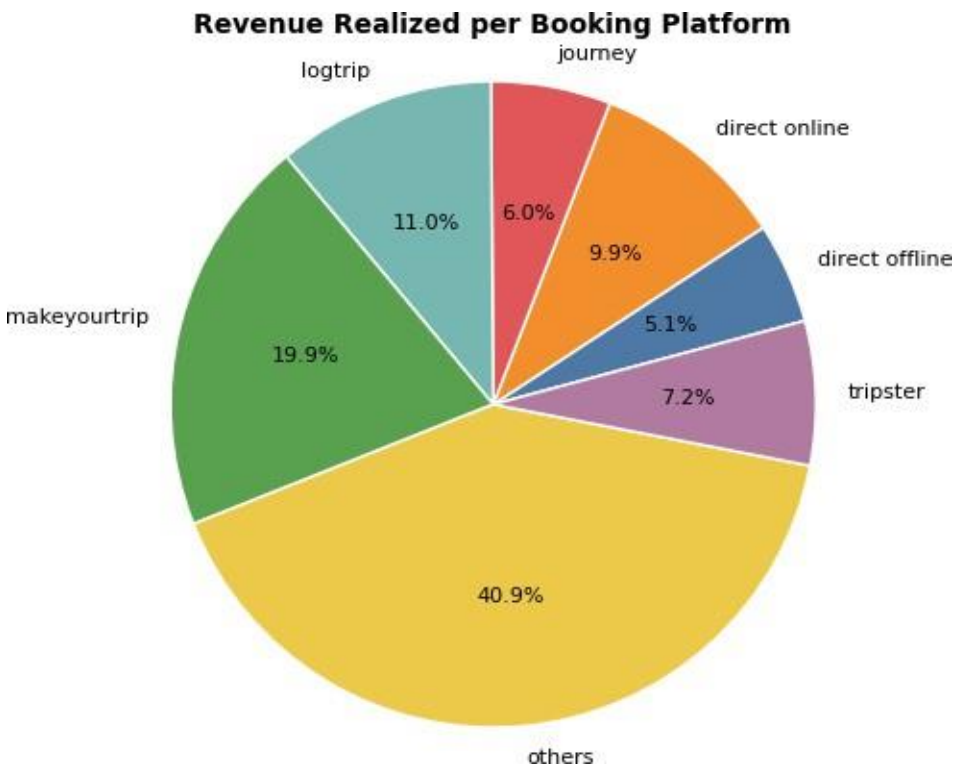


3. Print a pie chart of revenue realized per booking platform

[90] :
```
colors = ['#4E79A7', '#F28E2B', '#E15759', '#76B7B2',
          '#59A14F', '#EDC949', '#B07AA1']

(df.groupby('booking_platform')["revenue_realized"].sum()
   .plot(kind='pie',
         autopct='%1.1f%%',
         startangle=15,
         colors=colors,
         wedgeprops={'edgecolor': 'white'},textprops={'fontsize': 8}
        ))

plt.ylabel("")
plt.title("Revenue Realized per Booking Platform",fontsize=10,
 fontweight='bold')
plt.axis('equal')
plt.show()
```


Revenue Realized per Booking Platform
journey
logtrip
direct online
direct offline
makeyourtrip
tripster
others
6.0%
9.9%
5.1%
7.2%
40.9%
19.9%
11.0%


THANK
YOU