AtliQ Hotels Data Analysis Project

In [ ]:
```
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 dataframe

In [ ]:
```
df_bookings = pd.read_csv("D:\\365datascience\\Python\\codebasics_python\\3_project_hospitality_analysis\\datasets\\
df_bookings.head(6)
```

Out[ ]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	1.0	
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	5.0	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	NaN	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	

Explore bookings data

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

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

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

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

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

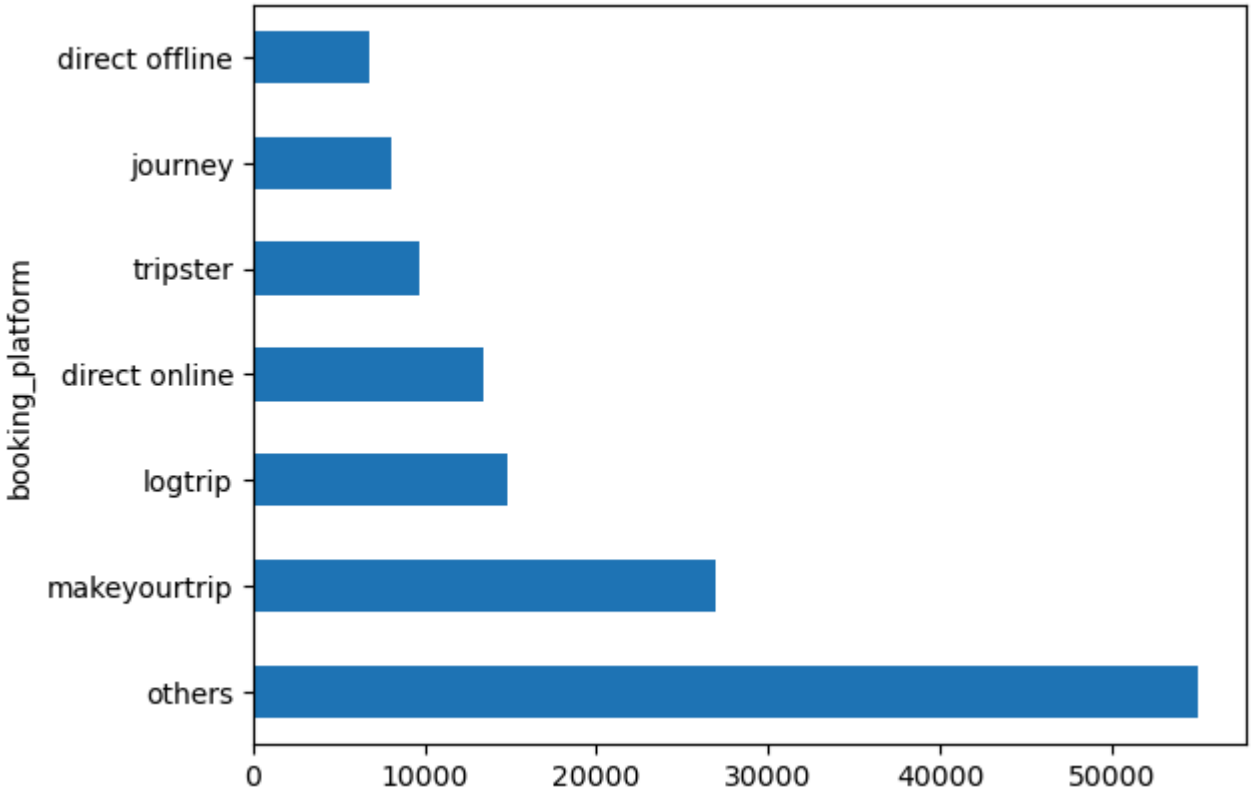
Out[ ]:
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

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

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

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

Out[ ]:
```
<Axes: ylabel='booking_platform'>
```



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

Out[ ]:

	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 [ ]:
```
df_date = pd.read_csv("D:\\365datascience\\Python\\codebasics_python\\3_project_hospitality_analysis\\datasets\\dim_
df_hotels = pd.read_csv("D:\\365datascience\\Python\\codebasics_python\\3_project_hospitality_analysis\\datasets\\di
df_rooms = pd.read_csv("D:\\365datascience\\Python\\codebasics_python\\3_project_hospitality_analysis\\datasets\\dim
df_agg_bookings = pd.read_csv("D:\\365datascience\\Python\\codebasics_python\\3_project_hospitality_analysis\\datase
```

In [ ]:
```
df_hotels.shape
```

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

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

Out[ ]:

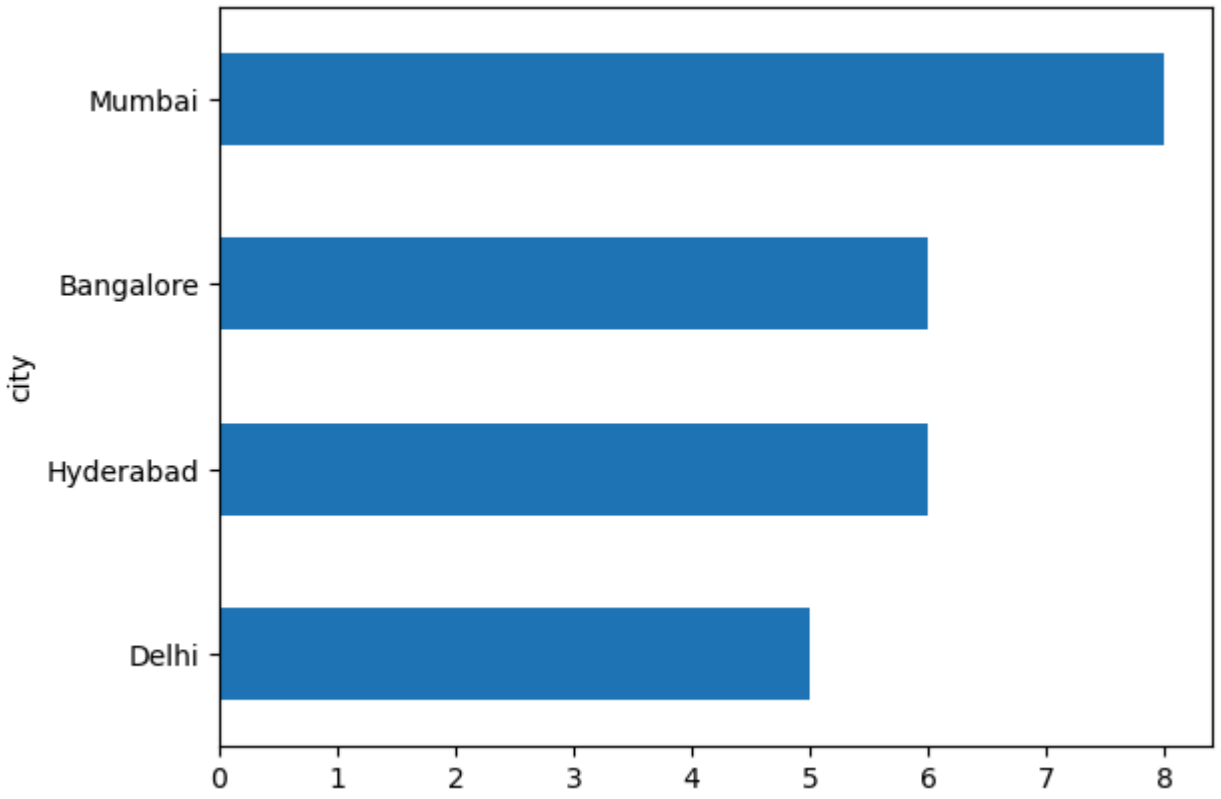
	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi
3	16561	Atliq Blu	Luxury	Delhi

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

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

In [ ]:
```
df_hotels["city"].value_counts().sort_values().plot(kind='barh')
```

Out[ ]:
```
<Axes: ylabel='city'>
```



Exercise: Explore aggregate bookings

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

Out[ ]:

	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

Exercise-1. Find out unique property ids in aggregate bookings dataset

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

Out[ ]:
```
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 [ ]:
```
"""
Cell generated by Data Wrangler.
"""
def clean_data(df_agg_bookings):
    # Performed 1 aggregation grouped on column: 'property_id'
    df_agg_bookings = df_agg_bookings.groupby(['property_id']).agg(successful_bookings_sum=('successful_bookings',
    return df_agg_bookings

df_agg_bookings_clean = clean_data(df_agg_bookings.copy())
df_agg_bookings_clean
```

Out[ ]:

	property_id	successful_bookings_sum
0	16558	3153
1	16559	7338
2	16560	4693
3	16561	4418
4	16562	4820
5	16563	7211
6	17558	5053
7	17559	6142
8	17560	6013
9	17561	5183
10	17562	3424
11	17563	6337
12	17564	3982
13	18558	4475
14	18559	5256
15	18560	6638
16	18561	6458
17	18562	7333
18	18563	4737

	property_id	successful_bookings_sum
19	19558	4400
20	19559	4729
21	19560	6079
22	19561	5736
23	19562	5812
24	19563	5413

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

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

Out[ ]:

	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 [ ]:
```
df_agg_bookings
```

Out[ ]:

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

9200 rows × 5 columns

In [ ]:

```
"""
Cell generated by Data Wrangler.
"""
def clean_data(df_agg_bookings):
    # Filter rows based on column: 'capacity'
    df_agg_bookings = df_agg_bookings[df_agg_bookings['capacity'] == 50]
    return df_agg_bookings

df_agg_bookings_clean = clean_data(df_agg_bookings.copy())
df_agg_bookings_clean
```

Out[ ]:

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

92 rows × 5 columns

==> 2. Data Cleaning

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

Out[ ]:

	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 [ ]:
```
df_bookings[df_bookings["no_guests"]<=0]
```

Out[ ]:

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

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

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

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

In [ ]:
```
df_bookings=df_bookings[df_bookings["no_guests"]>0]
df_bookings.head()
```

Out[ ]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	5.0	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	

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

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

(2) Outlier removal in revenue generated

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

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

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

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

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

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

lower_limit, higher_limit
```

Out[ ]:
```
(-263742.4278566132, 294498.50173198653)
```

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

Out[ ]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat

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

Out[ ]:

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

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

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

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

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

Out[ ]:
```
33479.3586618449
```

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

Out[ ]:

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

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

Out[ ]:
```
count    16071.000000
mean     27465.457034
std       6310.403418
min      19000.000000
25%      19000.000000
50%      28500.000000
75%      32300.000000
max      45220.000000
Name: revenue_generated, dtype: float64
```

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

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

Out[ ]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_stat

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

Out[ ]:
```
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 [ ]:
```
df_bookings[df_bookings.isna()].count()
```

Out[ ]:
```
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        0
booking_status       0
revenue_generated    0
revenue_realized     0
dtype: int64
```

In [ ]:
```
df_bookings
```

Out[ ]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_giv
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	N
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	N
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	N
...	...	...	...	...	...	...	...	...	
134584	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	2.0	RT4	others	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	1.0	RT4	makeyourtrip	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	1.0	RT4	tripster	N
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	2.0	RT4	logtrip	
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	2.0	RT4	makeyourtrip	N

134573 rows × 12 columns

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

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

In [ ]:
```
df_agg_bookings.isnull
df_agg_bookings
```

Out[ ]:

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

9200 rows × 5 columns

In [ ]:
```
"""
Cell generated by Data Wrangler.
"""
def clean_data(df_agg_bookings):
    # Replace missing values with the median of each column in: 'capacity'
    df_agg_bookings = df_agg_bookings.fillna({'capacity': df_agg_bookings['capacity'].median()})
    return df_agg_bookings
```

```
df_agg_bookings = clean_data(df_agg_bookings.copy())
df_agg_bookings
```

Out[ ]:

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

9200 rows × 5 columns

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

Out[ ]:
```
(9200, 5)
```

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

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

Out[ ]:

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

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

Out[ ]:
```
(9200, 5)
```

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

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

==> 3. Data Transformation

Create occupancy percentage column

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

Out[ ]:

	property_id	check_in_date	room_category	successful_bookings	capacity
0	16559	1-May-22	RT1	25	30.0
1	19562	1-May-22	RT1	28	30.0
2	19563	1-May-22	RT1	23	30.0
4	16558	1-May-22	RT1	18	19.0
5	17560	1-May-22	RT1	28	40.0

In [ ]:
```
df_agg_bookings["occ_pct"]= df_agg_bookings.apply(lambda x: x["successful_bookings"]/x["capacity"], axis=1)
df_agg_bookings.head()
```

```
C:\Users\ankit\AppData\Local\Temp\ipykernel_54140\4018569948.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_agg_bookings["occ_pct"]= df_agg_bookings.apply(lambda x: x["successful_bookings"]/x["capacity"], axis=1)
```

Out[ ]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
0	16559	1-May-22	RT1	25	30.0	0.833333
1	19562	1-May-22	RT1	28	30.0	0.933333
2	19563	1-May-22	RT1	23	30.0	0.766667
4	16558	1-May-22	RT1	18	19.0	0.947368
5	17560	1-May-22	RT1	28	40.0	0.700000

You can use following approach to get rid of SettingWithCopyWarning

In [ ]:
```
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[ ]:

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

Out[ ]:

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

Out[ ]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	NaN	

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

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

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

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

==> 4. Insights Generation

---

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

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

Out[ ]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
0	16559	1-May-22	RT1	25	30.0	83.33
1	19562	1-May-22	RT1	28	30.0	93.33
2	19563	1-May-22	RT1	23	30.0	76.67
4	16558	1-May-22	RT1	18	19.0	94.74
5	17560	1-May-22	RT1	28	40.0	70.00

In [ ]:
```
df_agg_bookings.groupby("room_category")["occ_pct"].mean().round(2)
```

Out[ ]:
```
room_category
RT1    57.89
RT2    58.01
RT3    58.03
RT4    59.28
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 [ ]:
```
df_rooms
```

Out[ ]:

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

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

Out[ ]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_id	room_class
0	16559	1-May-22	RT1	25	30.0	83.33	RT1	Standard
1	19562	1-May-22	RT1	28	30.0	93.33	RT1	Standard
2	19563	1-May-22	RT1	23	30.0	76.67	RT1	Standard
3	16558	1-May-22	RT1	18	19.0	94.74	RT1	Standard
4	17560	1-May-22	RT1	28	40.0	70.00	RT1	Standard

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

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

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

Out[ ]:

	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

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

Out[ ]:
```
57.88964285714285
```

2. Print average occupancy rate per city

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

Out[ ]:

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

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

Out[ ]:

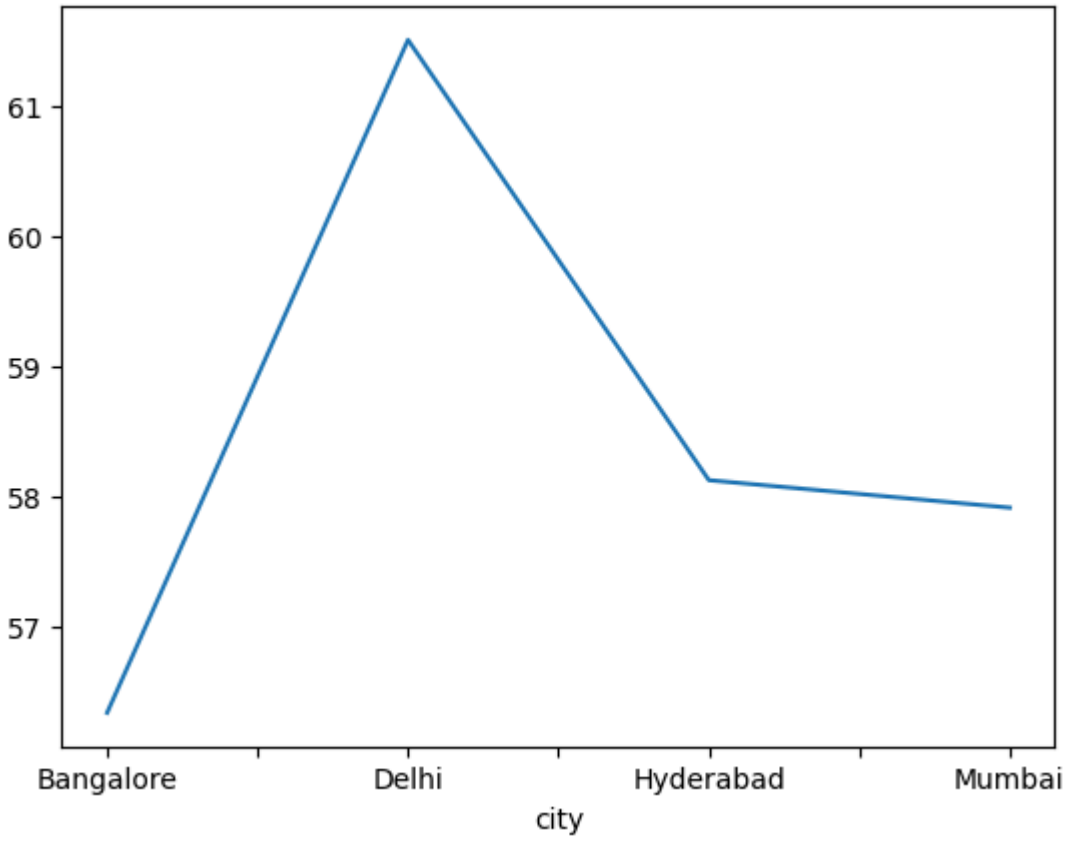
	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city
0	16559	1-May-22	RT1	25	30.0	83.33	Standard	Atliq Exotica	Luxury	Mumbai
1	19562	1-May-22	RT1	28	30.0	93.33	Standard	Atliq Bay	Luxury	Bangalore
2	19563	1-May-22	RT1	23	30.0	76.67	Standard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

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

Out[ ]:
```
city
Bangalore    56.332376
Delhi        61.507341
Hyderabad    58.120652
Mumbai       57.909181
Name: occ_pct, dtype: float64
```

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

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



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

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

Out[ ]:

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

Out[ ]:

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

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

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

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

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

Out[ ]:

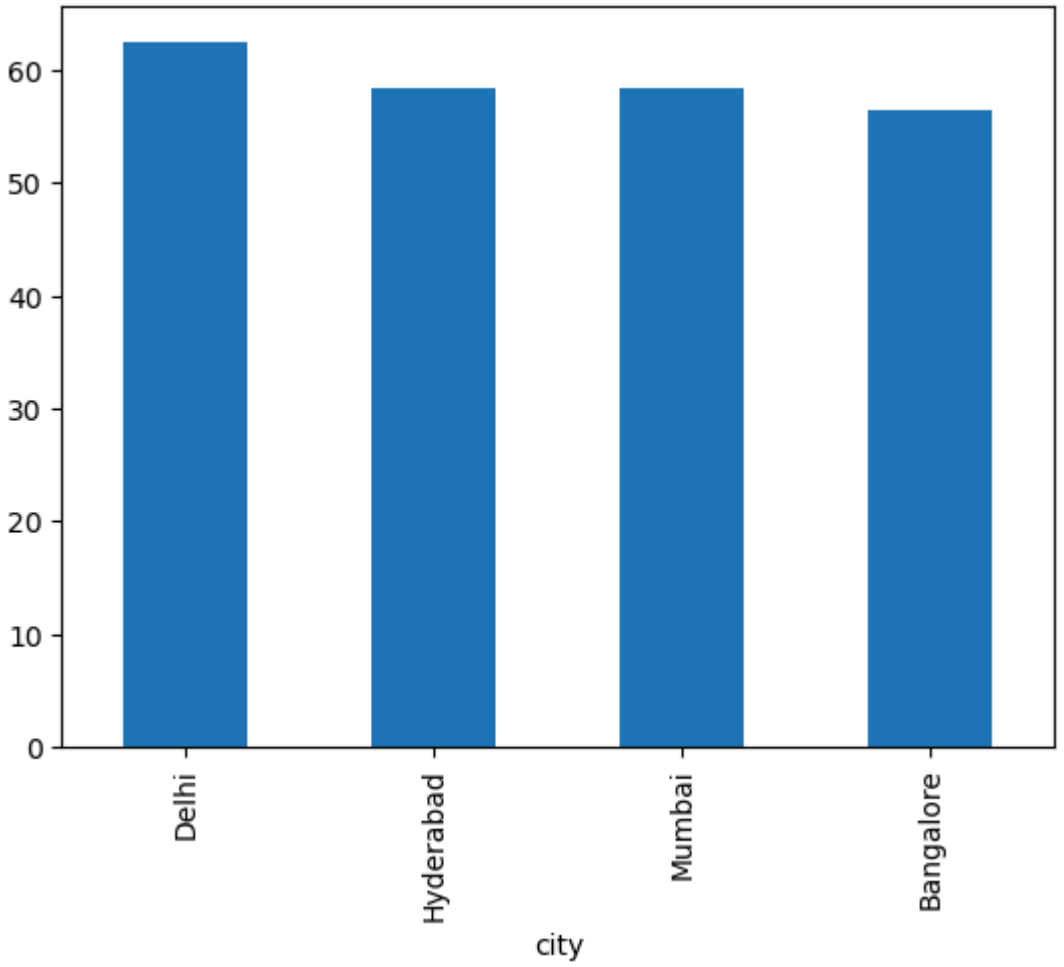
	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	da
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury	Mumbai	1 Ju 2
2201	19562	10-Jun-22	RT1	19	30.0	63.33	Standard	Atliq Bay	Luxury	Bangalore	1 Ju 2
2202	19563	10-Jun-22	RT1	17	30.0	56.67	Standard	Atliq Palace	Business	Bangalore	1 Ju 2
2203	17558	10-Jun-22	RT1	9	19.0	47.37	Standard	Atliq Grands	Luxury	Mumbai	1 Ju 2

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

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

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

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



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

In [ ]:
```
df_august = pd.read_csv("D:\\365datascience\\Python\\codebasics_python\\3_project_hospitality_analysis\\datasets\\ne
df_august.head(3)
```

Out[ ]:

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

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

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

Out[ ]:
```
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 [ ]:
```
df_august.shape
```

Out[ ]:
```
(7, 13)
```

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

Out[ ]:
```
(6497, 14)
```

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

Out[ ]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	da
6494	17558	31-Jul-22	RT4	3	6.0	50.0	Presidential	Atliq Grands	Luxury	Mumbai	3 J
6495	19563	31-Jul-22	RT4	3	6.0	50.0	Presidential	Atliq Palace	Business	Bangalore	3 J
6496	17561	31-Jul-22	RT4	3	4.0	75.0	Presidential	Atliq Blu	Luxury	Mumbai	3 J
6497	16559	01-Aug-22	RT1	30	30.0	NaN	Standard	Atliq Exotica	Luxury	Mumbai	Na
6498	19562	01-Aug-22	RT1	21	30.0	NaN	Standard	Atliq Bay	Luxury	Bangalore	Na
6499	19563	01-Aug-22	RT1	23	30.0	NaN	Standard	Atliq Palace	Business	Bangalore	Na
6500	19558	01-Aug-22	RT1	30	40.0	NaN	Standard	Atliq Grands	Luxury	Bangalore	Na
6501	19560	01-Aug-22	RT1	20	26.0	NaN	Standard	Atliq City	Business	Bangalore	Na
6502	17561	01-Aug-22	RT1	18	26.0	NaN	Standard	Atliq Blu	Luxury	Mumbai	Na
6503	17564	01-Aug-22	RT1	10	16.0	NaN	Standard	Atliq Seasons	Business	Mumbai	Na

In [ ]:
```
latest_df.shape
```

Out[ ]:
```
(6504, 15)
```

6. Print revenue realized per city

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

Out[ ]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	NaN	

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

Out[ ]:

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

Out[ ]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	
3	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	
4	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	NaN	

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

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

7. Print month by month revenue

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

Out[ ]:

	date	mmm yy	week no	day_type
0	01-May-22	May 22	W 19	weekend
1	02-May-22	May 22	W 19	weekeday
2	03-May-22	May 22	W 19	weekeday
3	04-May-22	May 22	W 19	weekeday
4	05-May-22	May 22	W 19	weekeday

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

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

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

Out[ ]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	
3	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	
4	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	NaN	

In [ ]:
```
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 [ ]:
```
df_date = df_date.astype({'date': 'datetime64[ns]'})
df_date.head()
```

Out[ ]:

	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
3	2022-05-04	May 22	W 19	weekeday
4	2022-05-05	May 22	W 19	weekeday

In [ ]:
```
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-null  object 
dtypes: float64(2), int64(3), object(10)
memory usage: 15.4+ MB
```

In [ ]:
```
# Change column type to datetime64[ns] for column: 'check_in_date'
df_bookings_all = df_bookings_all.astype({'check_in_date': 'datetime64[ns]'})
```

```
df_bookings_all.head()
```

Out[ ]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	b
0	May012216558RT12	16558	30-04-22	2022-01-05	2/5/2022	2.0	RT1	others	NaN	
1	May012216558RT15	16558	27-04-22	2022-01-05	2/5/2022	4.0	RT1	direct online	5.0	
2	May012216558RT16	16558	1/5/2022	2022-01-05	3/5/2022	2.0	RT1	others	4.0	
3	May012216558RT17	16558	28-04-22	2022-01-05	6/5/2022	2.0	RT1	others	NaN	
4	May012216558RT18	16558	26-04-22	2022-01-05	3/5/2022	2.0	RT1	logtrip	NaN	

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

Out[ ]:

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

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

Out[ ]:
```
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 [ ]:
```
df_bookings_all.property_name.unique()
```

Out[ ]:
```
array(['Atliq Grands', 'Atliq Exotica', 'Atliq City', 'Atliq Blu',
       'Atliq Bay', 'Atliq Palace', 'Atliq Seasons'], dtype=object)
```

In [ ]:
```
df_bookings_all.groupby("property_name")["revenue_realized"].sum().round(2).sort_values()
```

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

Exercise-2 Print average rating per city

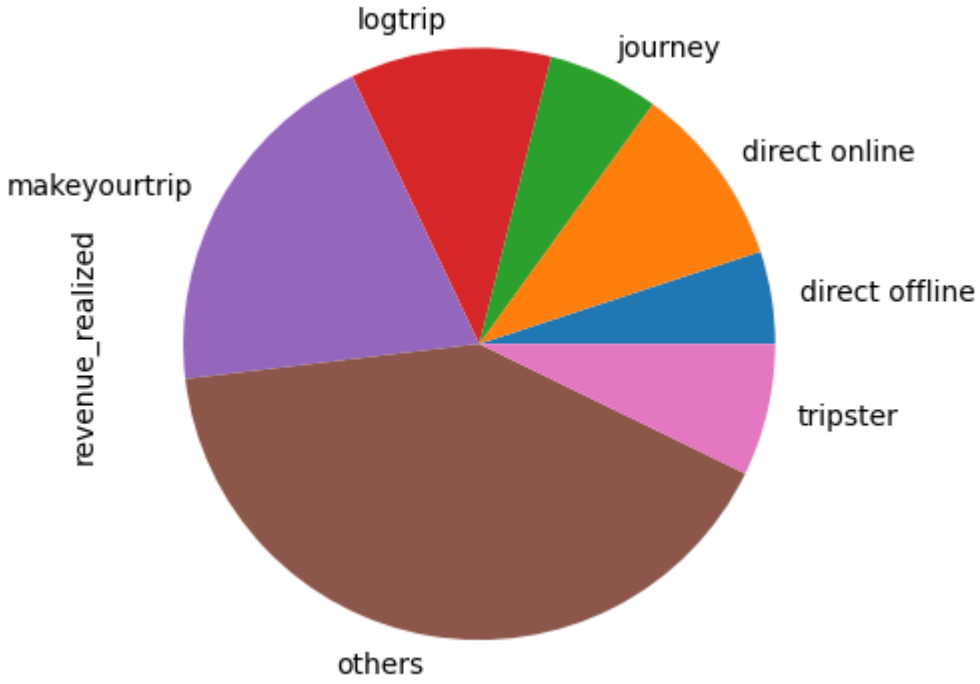
In [ ]:
```
df_bookings_all.groupby("city")["ratings_given"].mean().round(2)
```

Out[ ]:
```
city
Bangalore    3.40
Delhi        3.78
Hyderabad    3.66
Mumbai       3.64
Name: ratings_given, dtype: float64
```

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

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

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


logtrip
journey
direct online
makeyourtrip
direct offline
revenue_realized
tripster
others