AtliQ Grands Hospitality Data Analysis Project

In [1]:
```
import pandas as pd
```

---

1. Data Import and Data Exploration

Datasets

We have 5 csv files:

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

Read bookings data in a dataframe

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

Explore bookings data

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

Out[3]:

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

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

Out[4]:
```
(134590, 12)
```

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

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

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

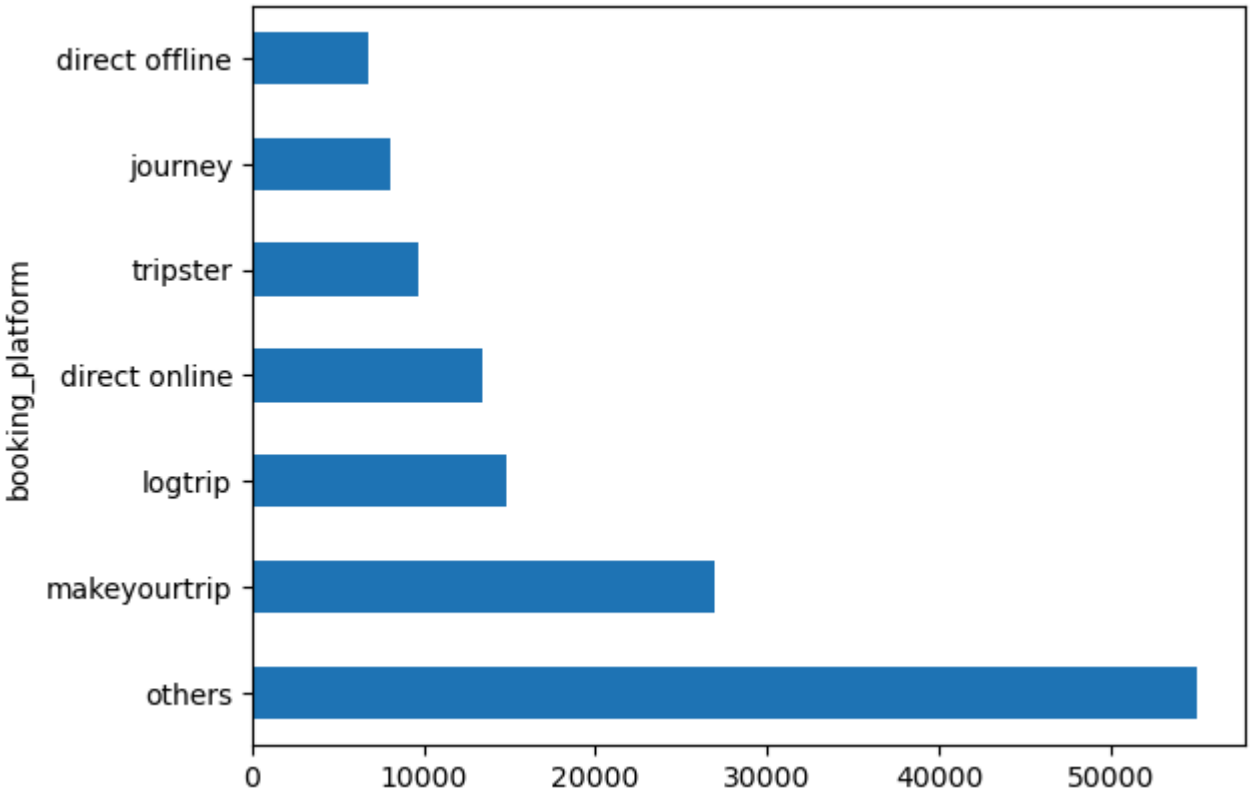
Out[6]:
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

In [48]:
```
df_bookings.booking_platform.value_counts()
```

Out[48]:
```
booking_platform
others            55066
makeyourtrip      26898
logtrip           14756
direct online     13379
tripster           9630
journey            8106
direct offline     6755
Name: count, dtype: int64
```

In [50]:
```
df_bookings.booking_platform.value_counts().plot(kind="barh")
```

Out[50]:
```
<Axes: ylabel='booking_platform'>
```



In [52]:
```
df_bookings.describe()
```

Out[52]:

	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

In [54]:
```
df_bookings.revenue_generated.min(),df_bookings.revenue_generated.max()
```

Out[54]:
```
(6500, 28560000)
```

Reading rest of the csv files

In [56]:
```
df_date = pd.read_csv("datasets/dim_date.csv")
df_hotels = pd.read_csv("datasets/dim_hotels.csv")
df_rooms = pd.read_csv("datasets/dim_rooms.csv")
df_agg_bookings = pd.read_csv("datasets/fact_aggregated_bookings.csv")
```

In [58]:
```
df_hotels.shape
```

Out[58]:
```
(25, 4)
```

In [60]:
```
df_hotels.head(4)
```

Out[60]:

	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 [62]:
```
df_hotels.category.value_counts()
```

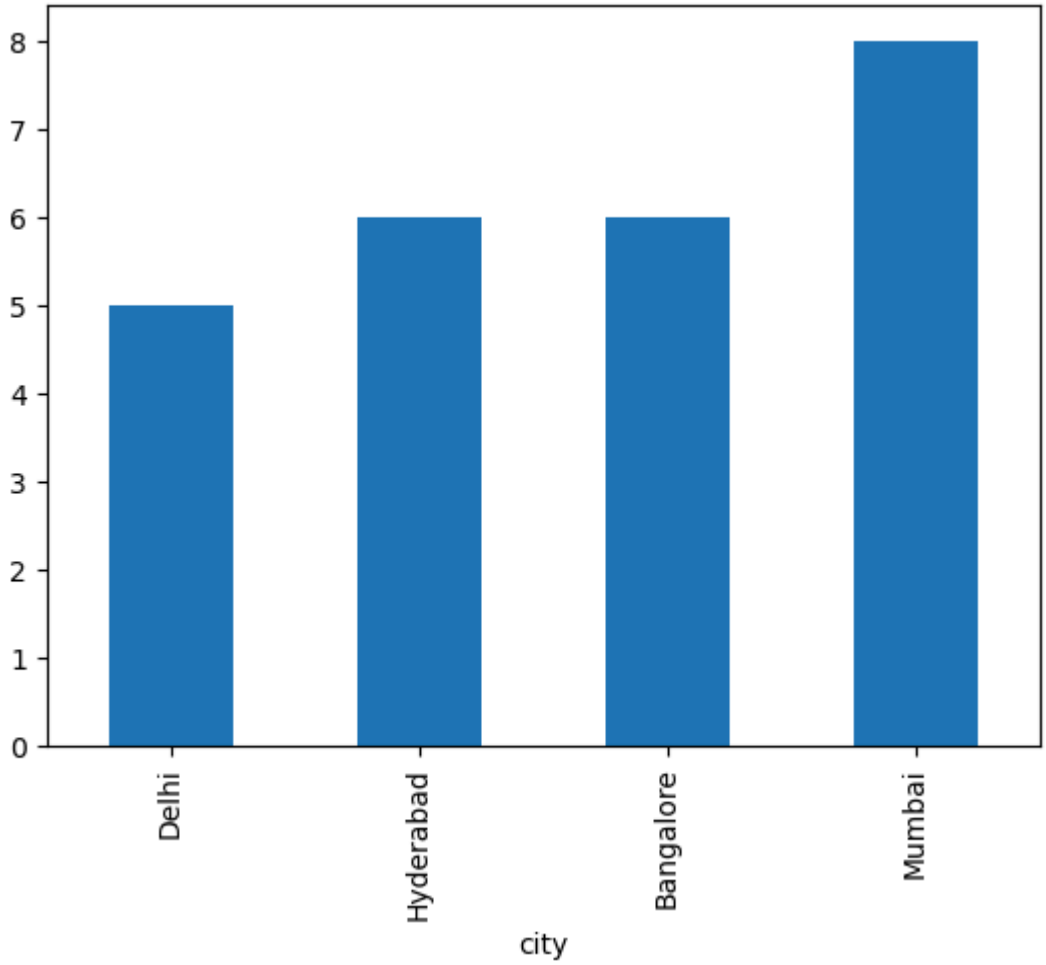
Out[62]:
```
category
Luxury      16
Business     9
Name: count, dtype: int64
```

In [64]:
```
df_hotels.city.value_counts().sort_values()
```

Out[64]:
```
city
Delhi        5
Hyderabad    6
Bangalore    6
Mumbai       8
Name: count, dtype: int64
```

In [66]:
```
df_hotels.city.value_counts().sort_values().plot(kind="bar")
```

Out[66]:
```
<Axes: xlabel='city'>
```



Explore Aggregate Bookings

In [68]:
```
df_agg_bookings.head(5)
```

Out[68]:

	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

Find out unique property ids in aggregate bookings dataset

In [70]:
```
df_agg_bookings.property_id.unique()
```

Out[70]:
```
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)
```

Find out total bookings per property_id

In [72]:
```
df_agg_bookings.groupby("property_id")["successful_bookings"].sum()
```

Out[72]:
```
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 [74]:
```
df_agg_bookings[df_agg_bookings.successful_bookings > df_agg_bookings.capacity]
```

Out[74]:

	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 [76]:
```
highest_capacity_properties = df_agg_bookings.sort_values(by="capacity", ascending=False)
highest_capacity_properties
```

Out[76]:

	property_id	check_in_date	room_category	successful_bookings	capacity
3128	17558	1-Jun-22	RT2	19	50.0
2128	17558	22-May-22	RT2	38	50.0
1728	17558	18-May-22	RT2	21	50.0
5828	17558	28-Jun-22	RT2	26	50.0
3928	17558	9-Jun-22	RT2	27	50.0
...	...	...	...	...	...
7475	19559	14-Jul-22	RT4	2	3.0
7476	16558	14-Jul-22	RT4	2	3.0
7375	19559	13-Jul-22	RT4	2	3.0
8	17561	1-May-22	RT1	22	NaN
14	17562	1-May-22	RT1	12	NaN

9200 rows × 5 columns

---

2. Data Cleaning

In [78]:
```
df_bookings.describe()
```

Out[78]:

	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

A. Clean Invalid Guests

In [80]:
```
df_bookings[df_bookings.no_guests<=0]
```

Out[80]:

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

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

In [82]:
```
df_bookings = df_bookings[df_bookings.no_guests>0]
df_bookings.shape
```

Out[82]:
```
(134578, 12)
```

B. Outlier removal in revenue generated

In [84]:
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

Out[84]:
```
(6500, 28560000)
```

In [86]:
```
avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.std()
avg, std
```

Out[86]:
```
(15378.036937686695, 93040.15493143328)
```

In [88]:
```
higher_limit = avg + 3*std
higher_limit
```

Out[88]:
```
294498.50173198653
```

In [90]:
```
lower_limit = avg - 3*std
lower_limit
```

Out[90]:
```
-263742.4278566132
```

In [92]:
```
df_bookings[df_bookings.revenue_generated<=0]
```

Out[92]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	bo

In [94]:
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

Out[94]:

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

In [96]:

```
df_bookings[df_bookings.revenue_generated<higher_limit]
df_bookings.shape
```

Out[96]:

```
(134578, 12)
```

In [98]:

```
df_bookings.revenue_realized.describe()
```

Out[98]:

```
count    134578.000000
mean      12696.011822
std        6927.841641
min        2600.000000
25%        7600.000000
50%       11700.000000
75%       15300.000000
max       45220.000000
Name: revenue_realized, dtype: float64
```

In [100…

```
higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std()
higher_limit
```

Out[100…

```
33479.53674501789
```

In [102…

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

Out[102…

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

1299 rows × 12 columns

One observation we can have in the above dataframe is that all rooms are RT4 which means they are presidential suites. 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 [104…
```
df_rooms
```

Out[104…

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

In [106…
```
df_bookings[df_bookings.room_category=="RT4"]
```

Out[106…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	roon
47	May012216558RT41	16558	26-04-22	1/5/2022	3/5/2022	2.0	
48	May012216558RT42	16558	27-04-22	1/5/2022	2/5/2022	1.0	
49	May012216558RT43	16558	29-04-22	1/5/2022	4/5/2022	2.0	
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	4.0	
138	May012216559RT42	16559	11/4/2022	1/5/2022	3/5/2022	2.0	
...	...	...	...	...	...	...	
134584	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	2.0	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	1.0	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	1.0	
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	2.0	
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	2.0	

16071 rows × 12 columns

In [108…
```
df_bookings[df_bookings.room_category=="RT4"].revenue_realized.describe()
```

Out[108…
```
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 [110…
```
# mean + 3*standard deviation
23439+3*9048
```

Out[110…
```
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 [112…
```
df_bookings[df_bookings.booking_id=="May012216558RT213"]
```

Out[112…

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	bo

In [114…
```
df_bookings.isnull().sum()
```

Out[114…
```
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        77899
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 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 [116…
```
df_agg_bookings.isnull().sum()
```

Out[116…
```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               2
dtype: int64
```

In [118…
```
df_agg_bookings[df_agg_bookings.capacity.isna()]
```

Out[118…

	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

In [120…
```
df_agg_bookings.capacity.median()
```

Out[120…
```
25.0
```

In [122…
```
df_agg_bookings.capacity.fillna(df_agg_bookings.capacity.median(), inplace=True)
df_agg_bookings.loc[[8,14]]
```

```
C:\Users\Tarun\AppData\Local\Temp\ipykernel_17116\4001445453.py:1: FutureWarning: A value is t
rying 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 interm
ediate object on which we are setting values always behaves as a copy.

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

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

Out[122…

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

In [124…

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

Out[124…

	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 [126…

```
df_agg_bookings.shape
```

Out[126…

```
(9200, 5)
```

In [128…

```
df_agg_bookings = df_agg_bookings[df_agg_bookings.successful_bookings<=df_agg_bookings.capaci
df_agg_bookings.shape
```

Out[128…

```
(9194, 5)
```

---

3. Data Transformation

Create occupancy percentage column

In [130…

```
df_agg_bookings.head()
```

Out[130…

	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 [134…

```
df_agg_bookings["occ_pct"] = df_agg_bookings["successful_bookings"]/df_agg_bookings["capacity
df_agg_bookings["occ_pct"]
```

Out[134…

```
0       0.833333
1       0.933333
2       0.766667
4       0.947368
5       0.700000
          ...
9195    0.722222
9196    0.722222
9197    0.500000
9198    0.500000
9199    0.750000
Name: occ_pct, Length: 9194, dtype: float64
```

In [136…

```
df_agg_bookings.head()
```

Out[136…

	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

In [138…

```
df_agg_bookings["occ_pct"] = df_agg_bookings["occ_pct"].apply(lambda x: round(x*100, 2))
df_agg_bookings.head(4)
```

Out[138…

	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

---

4. Insights Generation

---

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

In [140…

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

Out[140…

```
room_category
RT1    57.89
RT2    58.01
RT3    58.03
RT4    59.28
Name: occ_pct, dtype: float64
```

In [142…

```
df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_on="room_id")
df.tail(4)
```

Out[142…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_id	roo
9190	16559	31-Jul-22	RT4	13	18.0	72.22	RT4	Pre
9191	17558	31-Jul-22	RT4	3	6.0	50.00	RT4	Pre
9192	19563	31-Jul-22	RT4	3	6.0	50.00	RT4	Pre
9193	17561	31-Jul-22	RT4	3	4.0	75.00	RT4	Pre

In room_category, RT1, RT2 and so on may not make much sense. It would be great to have room categories defined such as Standard, Premium, etc.

In [144…

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

Out[144…

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

In [146…

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

Out[146…

	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 [148…

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

Out[148…

```
room_class
Elite           58.009756
Premium         58.028213
Presidential    59.277925
Standard        57.889643
Name: occ_pct, dtype: float64
```

2. Print average occupancy rate per city

In [150…

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

Out[150…

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

In [152…

```
df = pd.merge(df, df_hotels, on="property_id")
df.head(3)
```

Out[152…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	pro
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	

In [154…

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

Out[154…

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

3. When was the occupancy better? Weekdays or weekends?

In [156…

```
df.head(3)
```

Out[156…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	pro
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	

In [158…

```
df_date
```

Out[158…

	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
...	...	...	...	...
87	27-Jul-22	Jul 22	W 31	weekeday
88	28-Jul-22	Jul 22	W 31	weekeday
89	29-Jul-22	Jul 22	W 31	weekeday
90	30-Jul-22	Jul 22	W 31	weekend
91	31-Jul-22	Jul 22	W 32	weekend

92 rows × 4 columns

In [160…

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

Out[160…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	pro
0	19563	10-May-22	RT3	15	29.0	51.72	Premium	
1	18560	10-May-22	RT1	19	30.0	63.33	Standard	
2	19562	10-May-22	RT1	18	30.0	60.00	Standard	

In [162…

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

Out[162…

```
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 [164…

```
df["mmm yy"].unique()
```

Out[164…

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

In [166…

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

Out[166…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard
2201	19562	10-Jun-22	RT1	19	30.0	63.33	Standard
2202	19563	10-Jun-22	RT1	17	30.0	56.67	Standard
2203	17558	10-Jun-22	RT1	9	19.0	47.37	Standard

In [168…

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

Out[168…

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

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

In [170…

```
df_august = pd.read_csv("datasets/new_data_august.csv")
df_august
```

Out[170…

	property_id	property_name	category	city	room_category	room_class	check_in_date	mmm yy
0	16559	Atliq Exotica	Luxury	Mumbai	RT1	Standard	01-Aug-22	Aug-22
1	19562	Atliq Bay	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug-22
2	19563	Atliq Palace	Business	Bangalore	RT1	Standard	01-Aug-22	Aug-22
3	19558	Atliq Grands	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug-22
4	19560	Atliq City	Business	Bangalore	RT1	Standard	01-Aug-22	Aug-22
5	17561	Atliq Blu	Luxury	Mumbai	RT1	Standard	01-Aug-22	Aug-22
6	17564	Atliq Seasons	Business	Mumbai	RT1	Standard	01-Aug-22	Aug-22

In [172…

```
df_august.shape
```

Out[172…

```
(7, 13)
```

In [174…

```
df.shape
```

Out[174…

```
(6497, 14)
```

In [176…

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

Out[176…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class
6494	17558	31-Jul-22	RT4	3	6.0	50.0	Presidential
6495	19563	31-Jul-22	RT4	3	6.0	50.0	Presidential
6496	17561	31-Jul-22	RT4	3	4.0	75.0	Presidential
6497	16559	01-Aug-22	RT1	30	30.0	NaN	Standard
6498	19562	01-Aug-22	RT1	21	30.0	NaN	Standard
6499	19563	01-Aug-22	RT1	23	30.0	NaN	Standard
6500	19558	01-Aug-22	RT1	30	40.0	NaN	Standard
6501	19560	01-Aug-22	RT1	20	26.0	NaN	Standard
6502	17561	01-Aug-22	RT1	18	26.0	NaN	Standard
6503	17564	01-Aug-22	RT1	10	16.0	NaN	Standard

In [178…

```
latest_df.shape
```

Out[178…

```
(6504, 15)
```

6. Print revenue realized per city

In [180…

```
df_bookings.head(4)
```

Out[180…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cate
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	

In [182…

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

Out[182…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_cate
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	
1	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	
2	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	

In [184…

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

Out[184…

```
city
Bangalore    420383550
Delhi        294438788
Hyderabad    325179310
Mumbai       668602231
Name: revenue_realized, dtype: int64
```

Print revenue realized per hotel type

In [256…

```
df_bookings_all.property_name.unique()
```

Out[256…

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

In [259…

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

Out[259…

```
property_name
Atliq Seasons     66086735
Atliq Grands     211471234
Atliq Bay        260022118
Atliq Blu        260851922
Atliq City       285798439
Atliq Palace     304081863
Atliq Exotica    320291568
Name: revenue_realized, dtype: int64
```

Print average rating per city

In [264…

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

Out[264…

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

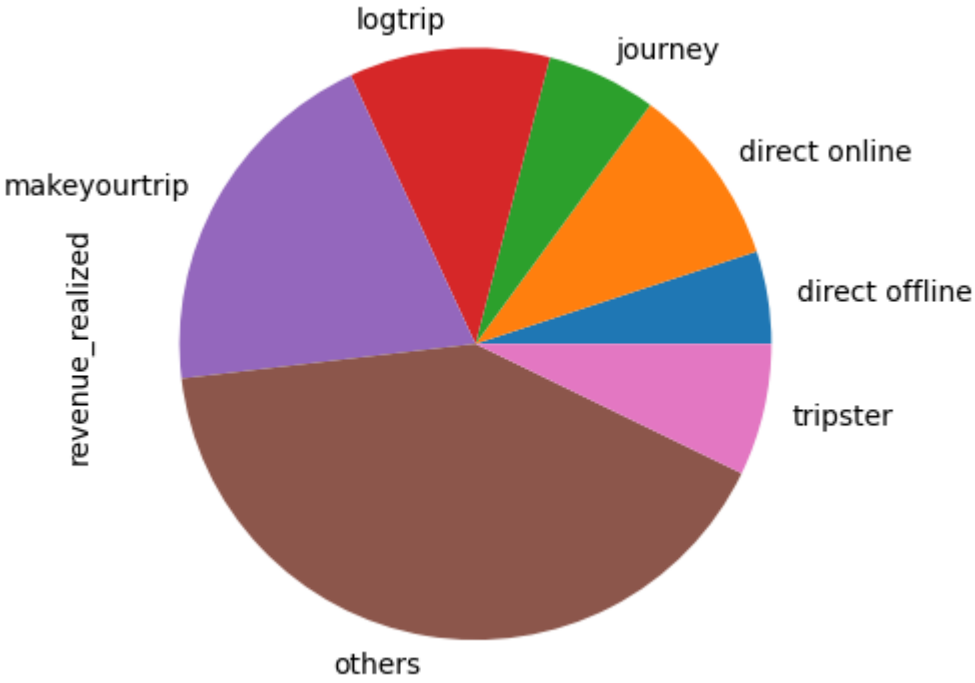
Print a pie chart of revenue realized per booking platform

In [271…

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

Out[271…

```
<Axes: ylabel='revenue_realized'>
```


logtrip
journey
direct online
direct offline
tripster
others
makeyourtrip
revenue_realized

In [ ]: