# AtliQ Hospitality Analysis

In [254…
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

---

## ==> 1. Data Import and Data Exploration

---

### Datasets

We have 5 csv file

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

**Read bookings data in a datagrame**

In [255…
```
df_bookings = pd.read_csv('datasets/fact_bookings.csv')
```

**Explore bookings data**

In [256…
```
df_bookings.head()
```

Out[256…

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

In [257…
```
df_bookings.shape
```

Out[257…
```
(134590, 12)
```

In [258…
```
df_bookings.room_category.unique()
```

Out[258…
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

In [259…
```
df_bookings.booking_platform.unique()
```

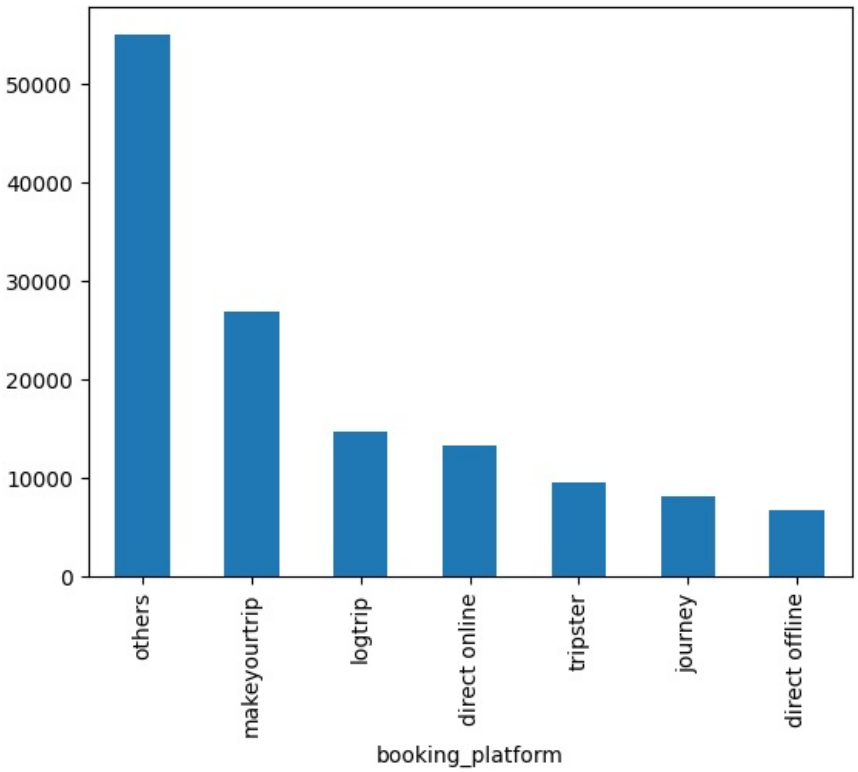
Out[259…
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

In [260…
```
df_bookings.booking_platform.value_counts()
```

Out[260…
```
booking_platform
others            55066
makeyourtrip      26898
logtrip           14756
direct online     13379
tripster           9630
journey            8106
direct offline     6755
Name: count, dtype: int64
```

In [261…
```
df_bookings.booking_platform.value_counts().plot(kind="bar")
```

Out[261…
```
<Axes: xlabel='booking_platform'>
```




In [262…
```
df_bookings.describe()
```

Out[262…

| | 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 [263…
```
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_agg_bookings.csv')
```

In [264…
```
df_hotels.shape
```

Out[264… (25, 4)

In [265…
```
df_hotels.head(3)
```

Out[265…

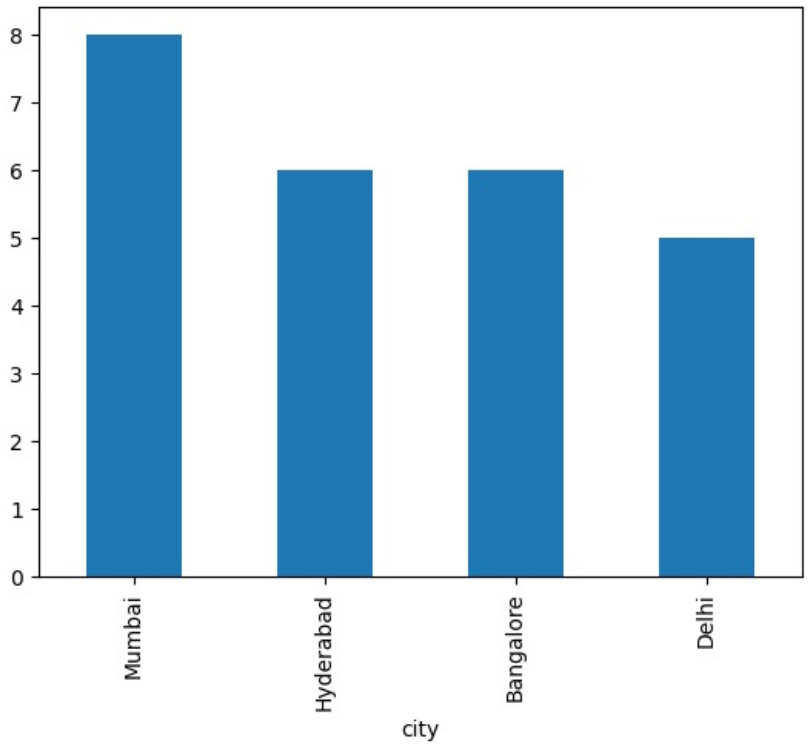
| | property_id | property_name | category | city |
|---|---|---|---|---|
| **0** | 16558 | Atliq Grands | Luxury | Delhi |
| **1** | 16559 | Atliq Exotica | Luxury | Mumbai |
| **2** | 16560 | Atliq City | Business | Delhi |

In [266…
```
df_hotels.category.value_counts()
```

Out[266…
```
category
Luxury      16
Business     9
Name: count, dtype: int64
```

In [267…
```
df_hotels.city.value_counts().plot(kind="bar")
```

Out[267… <Axes: xlabel='city'>




---

**Explore aggregate bookings**

---

In [268…
```
df_agg_bookings.head(3)
```

Out[268…

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

**Find out unique property ids in aggregate bookings dataset**

In [269…
```
df_agg_bookings["property_id"].unique()
```

Out[269…
```
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])
```

**Find out total bookings per property_id**

In [270…
```
df_agg_bookings.groupby('property_id')["successful_bookings"].sum()
```

Out[270…
```
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 [271…
```
# Convert 'check_in_date' from object (string) to datetime
df_agg_bookings["check_in_date"] = pd.to_datetime(df_agg_bookings["check_in_date"], format="%d-%b-%y")


df_agg_bookings["day_of_week"] = df_agg_bookings["check_in_date"].dt.day_name()

df_agg_bookings.groupby('day_of_week')[["successful_bookings", "capacity"]].sum()
```

Out[271…

| day_of_week | successful_bookings | capacity |
|---|---|---|
| **Friday** | 16908 | 32864.0 |
| **Monday** | 16925 | 32864.0 |
| **Saturday** | 24395 | 32864.0 |
| **Sunday** | 26049 | 35346.0 |
| **Thursday** | 16869 | 32864.0 |
| **Tuesday** | 16811 | 32864.0 |
| **Wednesday** | 16876 | 32864.0 |

In [272…
```
df_agg_bookings
```

Out[272…

| | property_id | check_in_date | room_category | successful_bookings | capacity | day_of_week |
|---|---|---|---|---|---|---|
| **0** | 16559 | 2022-05-01 | RT1 | 25 | 30.0 | Sunday |
| **1** | 19562 | 2022-05-01 | RT1 | 28 | 30.0 | Sunday |
| **2** | 19563 | 2022-05-01 | RT1 | 23 | 30.0 | Sunday |
| **3** | 17558 | 2022-05-01 | RT1 | 30 | 19.0 | Sunday |
| **4** | 16558 | 2022-05-01 | RT1 | 18 | 19.0 | Sunday |
| **...** | ... | ... | ... | ... | ... | ... |
| **9195** | 16563 | 2022-07-31 | RT4 | 13 | 18.0 | Sunday |
| **9196** | 16559 | 2022-07-31 | RT4 | 13 | 18.0 | Sunday |
| **9197** | 17558 | 2022-07-31 | RT4 | 3 | 6.0 | Sunday |
| **9198** | 19563 | 2022-07-31 | RT4 | 3 | 6.0 | Sunday |
| **9199** | 17561 | 2022-07-31 | RT4 | 3 | 4.0 | Sunday |

9200 rows × 6 columns

**Find out properties that have highest capacity**

In [273…
```
df_hotels.head()
```

Out[273…

| | 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 [274…
```
df_properties = pd.merge(df_agg_bookings, df_hotels, on="property_id")
df_properties.head(4)
```

Out[274…

| | property_id | check_in_date | room_category | successful_bookings | capacity | day_of_week | property_name | category | city |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 16559 | 2022-05-01 | RT1 | 25 | 30.0 | Sunday | Atliq Exotica | Luxury | Mumbai |
| 1 | 19562 | 2022-05-01 | RT1 | 28 | 30.0 | Sunday | Atliq Bay | Luxury | Bangalore |
| 2 | 19563 | 2022-05-01 | RT1 | 23 | 30.0 | Sunday | Atliq Palace | Business | Bangalore |
| 3 | 17558 | 2022-05-01 | RT1 | 30 | 19.0 | Sunday | Atliq Grands | Luxury | Mumbai |

In [275…
```
df_properties.groupby('property_name')[["capacity"]].sum().sort_values(by="capacity", ascending=False)
```

Out[275…

| property_name | capacity |
|---|---|
| Atliq Exotica | 40940.0 |
| Atliq Palace | 39376.0 |
| Atliq City | 39192.0 |
| Atliq Bay | 36596.0 |
| Atliq Blu | 35118.0 |
| Atliq Grands | 32384.0 |
| Atliq Seasons | 8924.0 |

---

## ==> 2. Data Cleaning

---

In [276…
```
df_bookings.describe()
```

Out[276…

| | 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 [277…
```
df_bookings[df_bookings.no_guests<=0]
```

Out[277…

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

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

In [278…
```
df_bookings = df_bookings[df_bookings.no_guests>0]
```

In [279…
```
df_bookings.shape
```

Out[279… (134578, 12)

**(2) Outlier removal in revenue generated**

In [280…
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

Out[280… (np.int64(6500), np.int64(28560000))

In [281…
```
df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median()
```

Out[281… (np.float64(15378.036937686695), np.float64(13500.0))

In [282…
```
avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.std()
```

In [283…
```
higher_limit = avg + 3*std
higher_limit
```

Out[283… np.float64(294498.50173207896)

In [284…
```
lower_limit = avg - 3*std
lower_limit
```

Out[284… np.float64(-263742.4278567056)

In [285…
```
df_bookings[df_bookings.revenue_generated<=0]
```

Out[285…

| | booking_id | property_id | booking_date | check_in_date | checkout_date | no_guests | room_category | booking_platform | ratings_give |
|---|---|---|---|---|---|---|---|---|---|

In [286…
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

Out[286…

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

In [287…
```
df_bookings = df_bookings[df_bookings.revenue_generated<=higher_limit]
df_bookings.shape
```

Out[287… (134573, 12)

In [288…
```
df_bookings.revenue_realized.describe()
```

Out[288…
```
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 [289…
```
higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std()
higher_limit
```

Out[289…
```
np.float64(33479.358661845814)
```

In [290…
```
df_bookings[df_bookings.revenue_realized>higher_limit]
```

Out[290…

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

1299 rows × 12 columns

One observation we can have in above dataframe is that all rooms are RT4 which means presidential suit. Now since RT4 is a luxurious room it is likely their rent will be higher. To make a fair analysis, we need to do data analysis only on RT4 room types

In [291…
```
df_bookings[df_bookings.room_category=="RT4"].revenue_realized.describe()
```

Out[291…
```
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 [292…
```
# mean + 3*standard deviation
23439+3*9048
```

Out[292…
```
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 [293…
```
df_bookings[df_bookings.booking_id=="May012216558RT213"]
```

Out[293…

| | booking_id | property_id | booking_date | check_in_date | checkout_date | no_guests | room_category | booking_platform | ratings_give |
|---|---|---|---|---|---|---|---|---|---|

In [294…
```
df_bookings.isnull().sum()
```

Out[294…
```
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 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 [295…
```
df_agg_bookings.isnull().sum()
```

Out[295…
```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               2
day_of_week            0
dtype: int64
```

In [296…
```
df_mean = round(df_agg_bookings["capacity"].mean(),2)
df_mean
```

Out[296…
```
np.float64(25.28)
```

In [297…
```
print(type(df_agg_bookings))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [298…
```
df_agg_bookings = pd.read_csv('datasets/fact_agg_bookings.csv')
df_agg_bookings
```

Out[298…

| | 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 [299…
```
df_agg_bookings["capacity"] = df_agg_bookings["capacity"].fillna(df_mean )

df_agg_bookings
```

Out[299…

| | 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 [300…
```
df_agg_bookings.isnull().sum()
```

Out[300…
```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               0
dtype: int64
```

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

In [359…
```
filtered_records = df_agg_bookings[df_agg_bookings["successful_bookings"] > df_agg_bookings["capacity"]]

filtered_records
```

Out[359…

| | property_id | check_in_date | room_category | successful_bookings | capacity | occ_pct |
|---|---|---|---|---|---|---|
| **3** | 17558 | 1-May-22 | RT1 | 30 | 19.0 | 157.89 |
| **12** | 16563 | 1-May-22 | RT1 | 100 | 41.0 | 243.90 |
| **4136** | 19558 | 11-Jun-22 | RT2 | 50 | 39.0 | 128.21 |
| **6209** | 19560 | 2-Jul-22 | RT1 | 123 | 26.0 | 473.08 |
| **8522** | 19559 | 25-Jul-22 | RT1 | 35 | 24.0 | 145.83 |
| **9194** | 18563 | 31-Jul-22 | RT4 | 20 | 18.0 | 111.11 |

---

## ==> 3. Data Transformation

---

**Create occupancy percentage column**

In [302…
```
df_agg_bookings.head(3)
```

Out[302…

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

In [303…
```
df_agg_bookings['occ_pct'] = df_agg_bookings.apply(lambda row: row['successful_bookings']/row['capacity'], axis
```

You can use following approach to get rid of SettingWithCopyWarning

In [304…
```
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[304…

| | 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 [305…
```
df_agg_bookings['occ_pct'] = df_agg_bookings['occ_pct'].apply(lambda x: round(x*100, 2))
df_agg_bookings.head(3)
```

Out[305…

| | 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 [306…
```
df_bookings
```

Out[306…

| | booking_id | property_id | booking_date | check_in_date | checkout_date | no_guests | room_category | booking_platform |
|---|---|---|---|---|---|---|---|---|
| **1** | May012216558RT12 | 16558 | 30-04-22 | 1/5/2022 | 2/5/2022 | 2.0 | RT1 | others |
| **4** | May012216558RT15 | 16558 | 27-04-22 | 1/5/2022 | 2/5/2022 | 4.0 | RT1 | direct online |
| **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 |
| **7** | May012216558RT18 | 16558 | 26-04-22 | 1/5/2022 | 3/5/2022 | 2.0 | RT1 | logtrip |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **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 | tripste |
| **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 |

134573 rows × 12 columns

In [307…
```
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

# Insights Generation

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

In [308…
```
df_agg_bookings.head(3)
```

Out[308…

| | 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 [309…
```
df_agg_bookings.groupby("room_category")["occ_pct"].mean()
```

Out[309…
```
room_category
RT1    58.232096
RT2    58.040278
RT3    58.028213
RT4    59.300461
Name: occ_pct, dtype: float64
```

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

In [310…
```
df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_on="room_id")
df.head(4)
```

Out[310…

| | property_id | check_in_date | room_category | successful_bookings | capacity | occ_pct | room_id | room_class |
|---|---|---|---|---|---|---|---|---|
| **0** | 16559 | 1-May-22 | RT1 | 25 | 30.0 | 83.33 | RT1 | Standard |
| **1** | 19562 | 1-May-22 | RT1 | 28 | 30.0 | 93.33 | RT1 | Standard |
| **2** | 19563 | 1-May-22 | RT1 | 23 | 30.0 | 76.67 | RT1 | Standard |
| **3** | 17558 | 1-May-22 | RT1 | 30 | 19.0 | 157.89 | RT1 | Standard |

In [311…
```
df.drop("room_id",axis=1, inplace=True)
df.head(4)
```

Out[311…

| | property_id | check_in_date | room_category | successful_bookings | capacity | occ_pct | room_class |
|---|---|---|---|---|---|---|---|
| **0** | 16559 | 1-May-22 | RT1 | 25 | 30.0 | 83.33 | Standard |
| **1** | 19562 | 1-May-22 | RT1 | 28 | 30.0 | 93.33 | Standard |
| **2** | 19563 | 1-May-22 | RT1 | 23 | 30.0 | 76.67 | Standard |
| **3** | 17558 | 1-May-22 | RT1 | 30 | 19.0 | 157.89 | Standard |

In [312…
```
df.groupby("room_class")["occ_pct"].mean()
```

Out[312…
```
room_class
Elite           58.040278
Premium         58.028213
Presidential    59.300461
Standard        58.232096
Name: occ_pct, dtype: float64
```

In [313…
```
df[df.room_class=="Standard"].occ_pct.mean()
```

Out[313…
```
np.float64(58.23209565217392)
```

**2. Print average occupancy rate per city**

In [314…
```
df_hotels.head(3)
```

Out[314…

| | 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 [315…
```
df = pd.merge(df, df_hotels, on="property_id")
df.head(3)
```

Out[315…

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

In [316…
```
df.groupby("city")["occ_pct"].mean()
```

Out[316…
```
city
Bangalore    56.594207
Delhi        61.606467
Hyderabad    58.144651
Mumbai       57.942632
Name: occ_pct, dtype: float64
```

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

In [317…
```
df_date.head(3)
```

Out[317…

| | 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 [318…
```
df = pd.merge(df, df_date, left_on="check_in_date", right_on="date")
df.head(3)
```

Out[318…

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

In [319…
```
df.groupby("day_type")["occ_pct"].mean().round(2)
```

Out[319…
```
day_type
weekeday    50.90
weekend     72.39
Name: occ_pct, dtype: float64
```

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

In [320…
```
df_june_22 = df[df["mmm yy"]=="Jun 22"]
df_june_22.head(4)
```

Out[320…

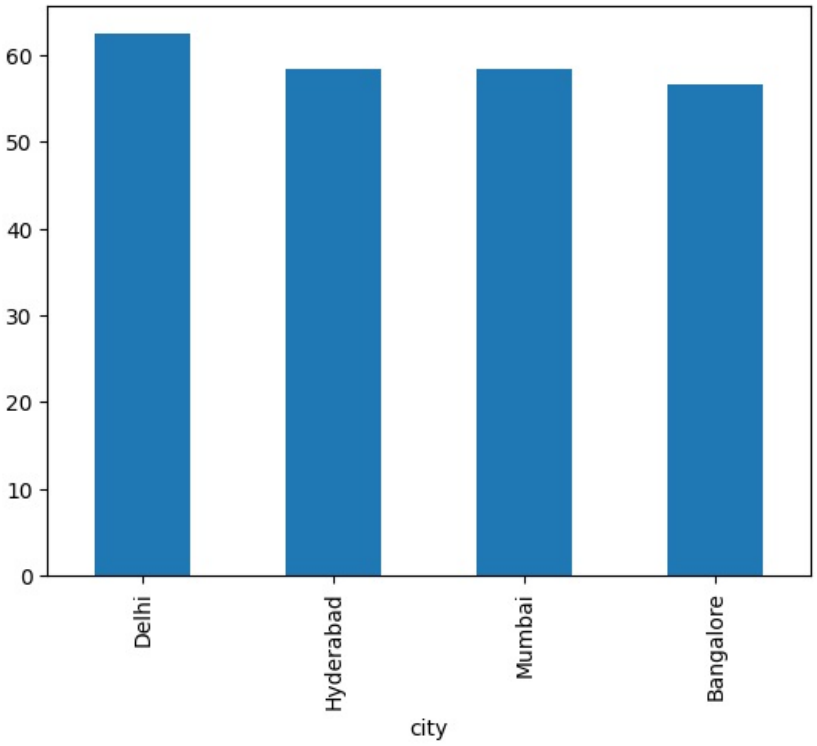
| | property_id | check_in_date | room_category | successful_bookings | capacity | occ_pct | room_class | property_name | category |
|---|---|---|---|---|---|---|---|---|---|
| **2200** | 16559 | 10-Jun-22 | RT1 | 20 | 30.0 | 66.67 | Standard | Atliq Exotica | Luxury |
| **2201** | 19562 | 10-Jun-22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

In [321…
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).sort_values(ascending=False)
```

Out[321…
```
city
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Bangalore    56.58
Name: occ_pct, dtype: float64
```

In [322…
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).sort_values(ascending=False).plot(kind="bar")
```

Out[322…
```
<Axes: xlabel='city'>
```




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

In [323…
```
df_august = pd.read_csv("datasets/new_data_august.csv")
df_august.head(3)
```

Out[323…

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

In [324…
```
df_august.columns
```

Out[324…
```
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 [325…
```
df.columns
```

Out[325…
```
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 [326…
```
df_august.shape
```

Out[326…
```
(7, 13)
```

In [327…
```
df.shape
```

Out[327…
```
(6500, 14)
```

In [328…
```
latest_df = pd.concat([df, df_august], ignore_index = True, axis = 0)
latest_df.tail(10)
```

Out[328…

| | property_id | check_in_date | room_category | successful_bookings | capacity | occ_pct | room_class | property_name | category |
|---|---|---|---|---|---|---|---|---|---|
| **6497** | 17558 | 31-Jul-22 | RT4 | 3 | 6.0 | 50.0 | Presidential | Atliq Grands | Luxury |
| **6498** | 19563 | 31-Jul-22 | RT4 | 3 | 6.0 | 50.0 | Presidential | Atliq Palace | Business |
| **6499** | 17561 | 31-Jul-22 | RT4 | 3 | 4.0 | 75.0 | Presidential | Atliq Blu | Luxury |
| **6500** | 16559 | 01-Aug-22 | RT1 | 30 | 30.0 | NaN | Standard | Atliq Exotica | Luxury |
| **6501** | 19562 | 01-Aug-22 | RT1 | 21 | 30.0 | NaN | Standard | Atliq Bay | Luxury |
| **6502** | 19563 | 01-Aug-22 | RT1 | 23 | 30.0 | NaN | Standard | Atliq Palace | Business |
| **6503** | 19558 | 01-Aug-22 | RT1 | 30 | 40.0 | NaN | Standard | Atliq Grands | Luxury |
| **6504** | 19560 | 01-Aug-22 | RT1 | 20 | 26.0 | NaN | Standard | Atliq City | Business |
| **6505** | 17561 | 01-Aug-22 | RT1 | 18 | 26.0 | NaN | Standard | Atliq Blu | Luxury |
| **6506** | 17564 | 01-Aug-22 | RT1 | 10 | 16.0 | NaN | Standard | Atliq Seasons | Business |

In [329…
```
latest_df.shape
```

Out[329… (6507, 15)

**6. Print revenue realized per city**

In [330…
```
df_bookings = pd.read_csv('datasets/fact_bookings.csv')
df_bookings
```

Out[330…

| | booking_id | property_id | booking_date | check_in_date | checkout_date | no_guests | room_category | booking_platform |
|---|---|---|---|---|---|---|---|---|
| **0** | May012216558RT11 | 16558 | 27-04-22 | 1/5/2022 | 2/5/2022 | -3.0 | RT1 | direct online |
| **1** | May012216558RT12 | 16558 | 30-04-22 | 1/5/2022 | 2/5/2022 | 2.0 | RT1 | others |
| **2** | May012216558RT13 | 16558 | 28-04-22 | 1/5/2022 | 4/5/2022 | 2.0 | RT1 | logtrip |
| **3** | May012216558RT14 | 16558 | 28-04-22 | 1/5/2022 | 2/5/2022 | -2.0 | RT1 | others |
| **4** | May012216558RT15 | 16558 | 27-04-22 | 1/5/2022 | 2/5/2022 | 4.0 | RT1 | direct online |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **134585** | Jul312217564RT46 | 17564 | 29-07-22 | 31-07-22 | 3/8/2022 | 1.0 | RT4 | makeyourtrip |
| **134586** | Jul312217564RT47 | 17564 | 30-07-22 | 31-07-22 | 1/8/2022 | -4.0 | RT4 | logtrip |
| **134587** | Jul312217564RT48 | 17564 | 30-07-22 | 31-07-22 | 2/8/2022 | 1.0 | RT4 | tripste |
| **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 |

134590 rows × 12 columns

In [331…
```
df_hotels = pd.read_csv('datasets/dim_hotels.csv')
df_hotels.head(3)
```

Out[331…

| | 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 [332…
```
df_bookings_all = pd.merge(df_bookings, df_hotels, on="property_id")
df_bookings_all
```

Out[332…

| | booking_id | property_id | booking_date | check_in_date | checkout_date | no_guests | room_category | booking_platform |
|---|---|---|---|---|---|---|---|---|
| **0** | May012216558RT11 | 16558 | 27-04-22 | 1/5/2022 | 2/5/2022 | -3.0 | RT1 | direct online |
| **1** | May012216558RT12 | 16558 | 30-04-22 | 1/5/2022 | 2/5/2022 | 2.0 | RT1 | others |
| **2** | May012216558RT13 | 16558 | 28-04-22 | 1/5/2022 | 4/5/2022 | 2.0 | RT1 | logtrip |
| **3** | May012216558RT14 | 16558 | 28-04-22 | 1/5/2022 | 2/5/2022 | -2.0 | RT1 | others |
| **4** | May012216558RT15 | 16558 | 27-04-22 | 1/5/2022 | 2/5/2022 | 4.0 | RT1 | direct online |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **134585** | Jul312217564RT46 | 17564 | 29-07-22 | 31-07-22 | 3/8/2022 | 1.0 | RT4 | makeyourtrip |
| **134586** | Jul312217564RT47 | 17564 | 30-07-22 | 31-07-22 | 1/8/2022 | -4.0 | RT4 | logtrip |
| **134587** | Jul312217564RT48 | 17564 | 30-07-22 | 31-07-22 | 2/8/2022 | 1.0 | RT4 | tripste |
| **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 |

134590 rows × 15 columns

In [333…
```
df_bookings_all.groupby("city")["revenue_realized"].sum()
```

Out[333…
```
city
Bangalore    420397050
Delhi        294500318
Hyderabad    325232870
Mumbai       668640991
Name: revenue_realized, dtype: int64
```

**7. Print month by month revenue**

In [334…
```
df_date = pd.read_csv('datasets/dim_date.csv')
df_date.head(3)
```

Out[334…

| | 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 [335…
```
df_date["mmm yy"].unique()
```

Out[335…
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

In [336…
```
df_bookings_all.head(3)
```

Out[336…

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

In [337…
```
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 [338…
```
df_date["date"] = pd.to_datetime(df_date["date"], errors="coerce")
df_date.head()
```

Out[338…

| | date | mmm yy | week no | day_type |
|---|---|---|---|---|
| 0 | 2022-05-01 | May 22 | W 19 | weekend |
| 1 | 2022-05-02 | May 22 | W 19 | weekday |
| 2 | 2022-05-03 | May 22 | W 19 | weekday |
| 3 | 2022-05-04 | May 22 | W 19 | weekday |
| 4 | 2022-05-05 | May 22 | W 19 | weekday |

In [339…
```
df_bookings_all.head()
```

Out[339…

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

In [340…
```
df_bookings_all["check_in_date"] = pd.to_datetime(df_bookings_all["check_in_date"], errors="coerce")
df_bookings_all.head()
```

Out[340…

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

In [341…
```
df_bookings_all.head()
```

Out[341…

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

In [342…
```
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on="check_in_date", right_on="date")
df_bookings_all.head(3)
```

Out[342…

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

In [343…
```
df_bookings_all.groupby("mmm yy")["revenue_realized"].sum()
```

Out[343…
```
mmm yy
Jul 22    60278496
Jun 22    52903014
May 22    60961428
Name: revenue_realized, dtype: int64
```

**Print revenue realized per hotel type**

In [344…
```
df_bookings_all.groupby("room_category")["revenue_realized"].sum()
```

Out[344…
```
room_category
RT1    31707494
RT2    56984850
RT3    47181288
RT4    38269306
Name: revenue_realized, dtype: int64
```

**Print average rating per city**

In [345…
```
# write your code here
round(df_bookings_all.groupby("city")["ratings_given"].mean(),2)
```

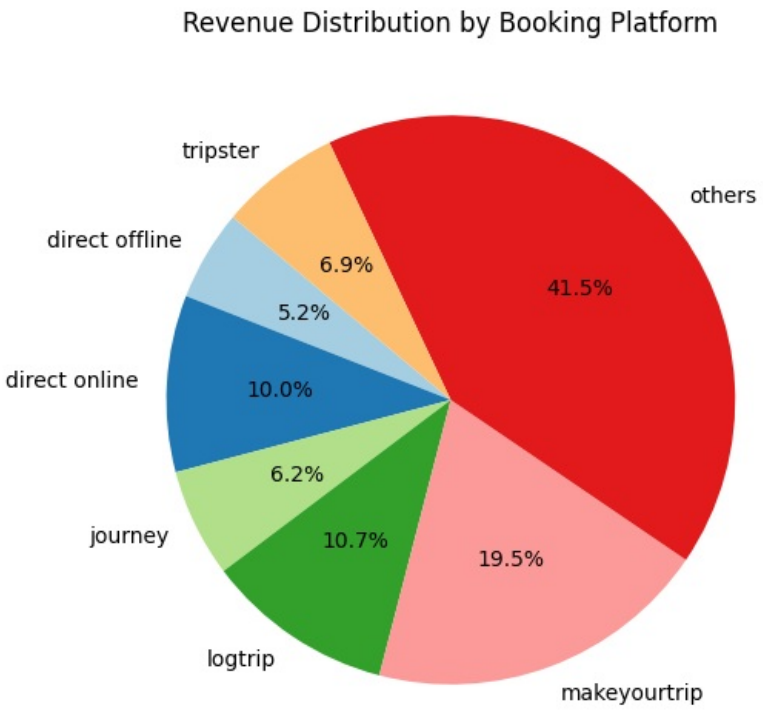
Out[345…
```
city
Bangalore    3.41
Delhi        3.79
Hyderabad    3.65
Mumbai       3.63
Name: ratings_given, dtype: float64
```

**Print a pie chart of revenue realized per booking platform**

In [346…
```
# write your code here
df_bookings_all.groupby("booking_platform")["revenue_realized"].sum()
```

Out[346…
```
booking_platform
direct offline     8986465
direct online     17488976
journey           10757858
logtrip           18605339
makeyourtrip      34034257
others            72310965
tripster          11959078
Name: revenue_realized, dtype: int64
```

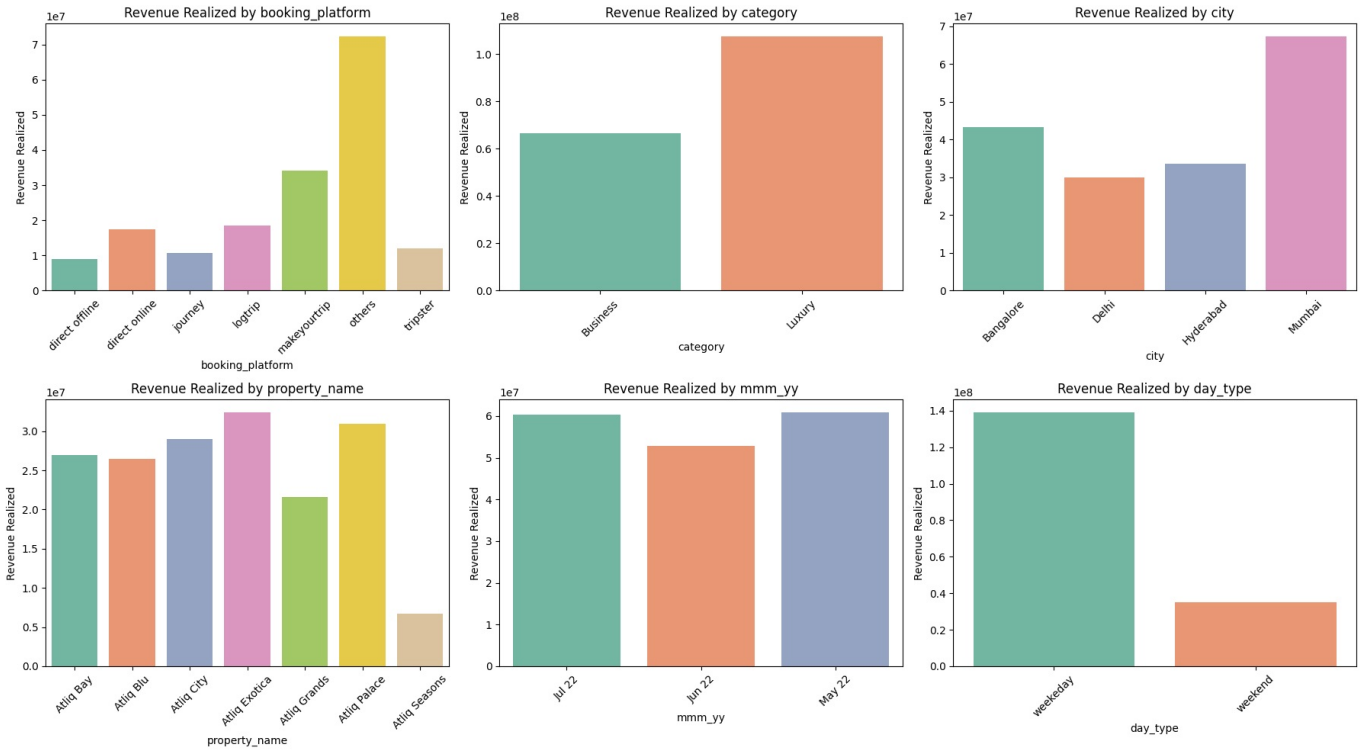
In [347…
```
import matplotlib.pyplot as plt

# Group data by booking platform and sum revenue
platform_revenue = df_bookings_all.groupby("booking_platform")["revenue_realized"].sum()

# Plot pie chart
plt.figure(figsize=(6, 6))
plt.pie(platform_revenue, labels=platform_revenue.index, autopct="%1.1f%%", startangle=140, colors=plt.cm.Paire
plt.title("Revenue Distribution by Booking Platform")
plt.show()
```




## Data Visualization By Revenue Realized

In [360…
```
import matplotlib.pyplot as plt
import seaborn as sns

# Grouping the data as per your requirements
```

```
grouped_data = {
    'booking_platform': df_bookings_all.groupby("booking_platform")["revenue_realized"].sum(),
    'category': df_bookings_all.groupby("category")["revenue_realized"].sum(),
    'city': df_bookings_all.groupby("city")["revenue_realized"].sum(),
    'property_name': df_bookings_all.groupby("property_name")["revenue_realized"].sum(),
    'mmm_yy': df_bookings_all.groupby('mmm yy')["revenue_realized"].sum(),
    'day_type': df_bookings_all.groupby('day_type')["revenue_realized"].sum(),
}

# Set up the grid for subplots
fig, axes = plt.subplots(2, 3, figsize=(18, 10))  # 2 rows, 3 columns

# List of group names and axes for plotting
group_names = list(grouped_data.keys())
axes = axes.flatten()

# Loop through each group and create a bar plot
for i, (group_name, data) in enumerate(grouped_data.items()):
    sns.barplot(x=data.index, y=data.values, ax=axes[i], palette='Set2')  # 'Set2' palette for color variation
    axes[i].set_title(f'Revenue Realized by {group_name}')
    axes[i].tick_params(axis='x', rotation=45)  # Rotate x-axis labels for better readability
    axes[i].set_xlabel(group_name)
    axes[i].set_ylabel('Revenue Realized')

# Adjust layout to make it neat
plt.tight_layout()

# Save the figure as a PNG file
plt.savefig('Revenue_realized_by_group.png', format='png')

# Show the plot
plt.show()
```

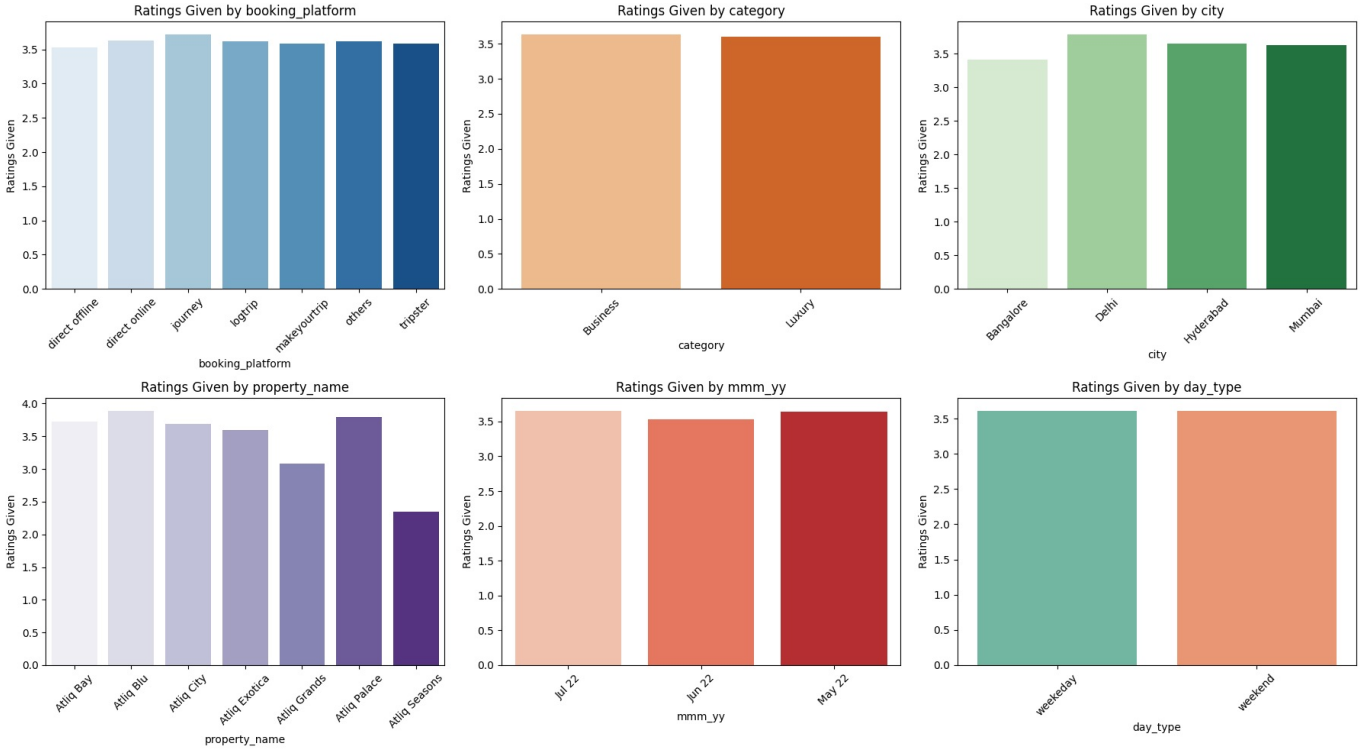



## Data Visualization By Ratings Given

In [361…

```
# Grouping the data as per your requirements
grouped_data = {
    'booking_platform': df_bookings_all.groupby("booking_platform")["ratings_given"].mean(),
    'category': df_bookings_all.groupby("category")["ratings_given"].mean(),
    'city': df_bookings_all.groupby("city")["ratings_given"].mean(),
    'property_name': df_bookings_all.groupby("property_name")["ratings_given"].mean(),
    'mmm_yy': df_bookings_all.groupby('mmm yy')["ratings_given"].mean(),
    'day_type': df_bookings_all.groupby('day_type')["ratings_given"].mean(),
}

# Define different color palettes for each plot
color_palettes = ['Blues', 'Oranges', 'Greens', 'Purples', 'Reds', 'Set2']

# Set up the grid for subplots
fig, axes = plt.subplots(2, 3, figsize=(18, 10))  # 2 rows, 3 columns

# List of group names and axes for plotting
```

```
group_names = list(grouped_data.keys())
axes = axes.flatten()

# Loop through each group and create a bar plot with a different color palette
for i, (group_name, data) in enumerate(grouped_data.items()):
    sns.barplot(x=data.index, y=data.values, ax=axes[i], palette=color_palettes[i])  # Use a different palette
    axes[i].set_title(f'Ratings Given by {group_name}')
    axes[i].tick_params(axis='x', rotation=45)  # Rotate x-axis labels for better readability
    axes[i].set_xlabel(group_name)
    axes[i].set_ylabel('Ratings Given')

# Adjust layout to make it neat
plt.tight_layout()

# Save the figure as a PNG file
plt.savefig('Ratings_given_by_group.png', format='png')

plt.show()
```




In [ ]: