AtliQ Hotels Data Analysis Project

```
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 datagrame

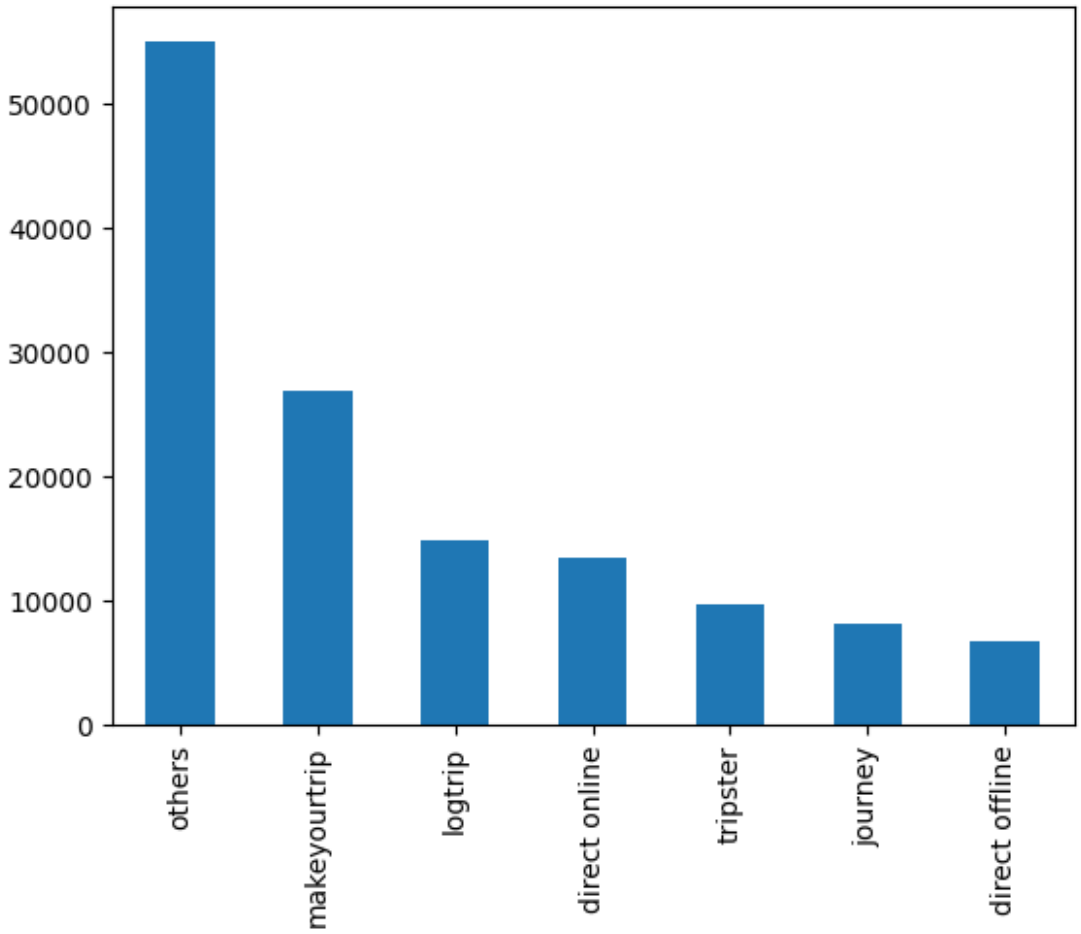
```
df_bookings = pd.read_csv('datasets/fact_bookings.csv')
```

Explore bookings data

```
df_bookings.head()
```

```
         booking_id  property_id booking_date check_in_date
checkout_date  \
0  May012216558RT11        16558     27-04-22      1/5/2022
2/5/2022
1  May012216558RT12        16558     30-04-22      1/5/2022
2/5/2022
2  May012216558RT13        16558     28-04-22      1/5/2022
4/5/2022
3  May012216558RT14        16558     28-04-22      1/5/2022
2/5/2022
4  May012216558RT15        16558     27-04-22      1/5/2022
2/5/2022

   no_guests room_category booking_platform  ratings_given
booking_status  \
0       -3.0           RT1    direct online            1.0    Checked
Out
1        2.0           RT1           others            NaN
Cancelled
2        2.0           RT1          logtrip            5.0    Checked
```

```
Out
3       -2.0           RT1           others             NaN
Cancelled
4        4.0           RT1    direct online             5.0    Checked
Out

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

df_bookings.shape

(134590, 12)

df_bookings.room_category.unique()

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

df_bookings.booking_platform.unique()

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

df_bookings.booking_platform.value_counts()

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

df_bookings.booking_platform.value_counts().plot(kind="bar")

<AxesSubplot: >
```




```
df_bookings.describe()
```

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

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

Read rest of the files

```
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')

df_hotels.shape
```

```
(25, 4)
```

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

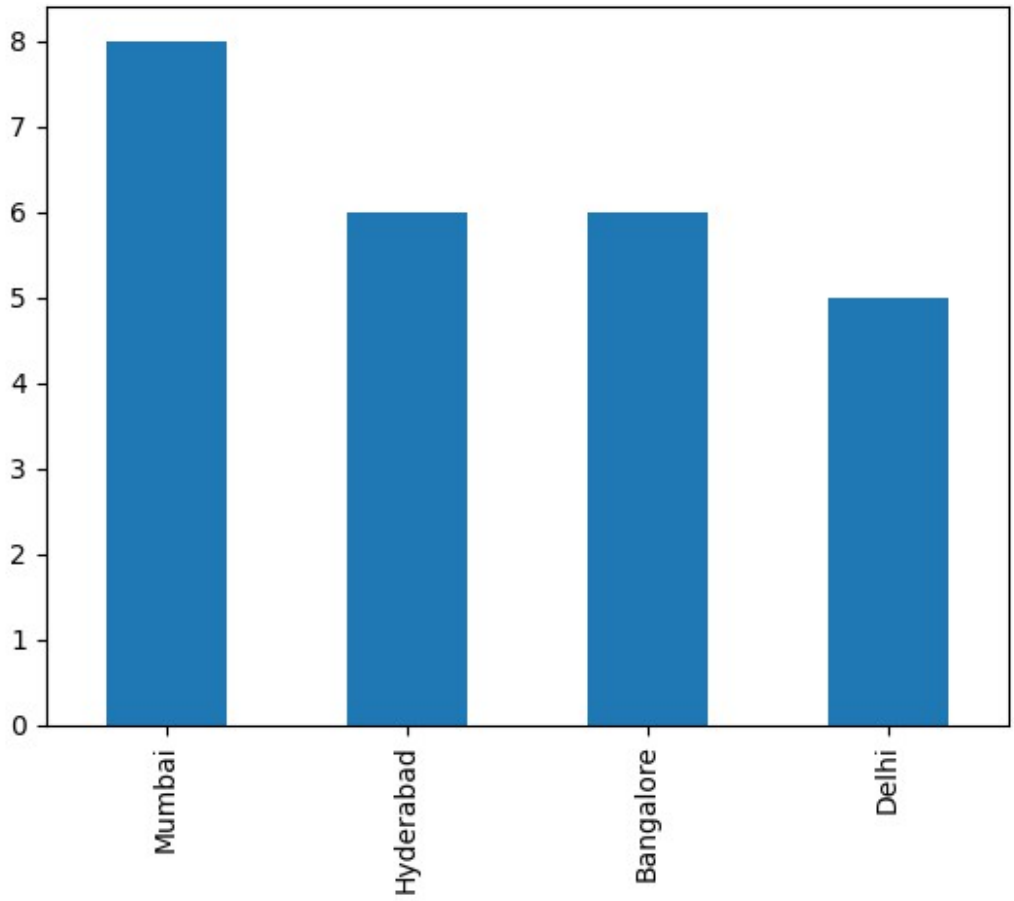
```
   property_id  property_name  category    city
0        16558   Atliq Grands    Luxury   Delhi
1        16559  Atliq Exotica    Luxury  Mumbai
2        16560     Atliq City  Business   Delhi
```

```
df_hotels.category.value_counts()
```

```
Luxury      16
Business     9
Name: category, dtype: int64
```

```
df_hotels.city.value_counts().plot(kind="bar")
```

```
<AxesSubplot: >
```



Exercise: Explore aggregate bookings ***

```
df_agg_bookings.head(3)
```

```
   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
```

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

```
df_agg_bookings.property_id.unique()
```

```
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

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

```
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
```

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

```
df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.ca
pacity]
```

```
      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

```
df_agg_bookings.capacity.max()

50.0

df_agg_bookings[df_agg_bookings.capacity==df_agg_bookings.capacity.max
()]

      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 x 5 columns]
```

==> 2. Data Cleaning

```
df_bookings.describe()

         property_id      no_guests  ratings_given  revenue_generated
\
count  134590.000000  134587.000000   56683.000000       1.345900e+05
```

```
mean    18061.113493       2.036170       3.619004       1.537805e+04

std      1093.055847       1.034885       1.235009       9.303604e+04

min     16558.000000     -17.000000       1.000000       6.500000e+03

25%     17558.000000       1.000000       3.000000       9.900000e+03

50%     17564.000000       2.000000       4.000000       1.350000e+04

75%     18563.000000       2.000000       5.000000       1.800000e+04

max     19563.000000       6.000000       5.000000       2.856000e+07


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

(1) Clean invalid guests

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

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

       checkout_date  no_guests room_category booking_platform
ratings_given  \
0           2/5/2022       -3.0           RT1    direct online
1.0
3           2/5/2022       -2.0           RT1           others
NaN
17924       14-05-22      -10.0           RT4    direct online
NaN
18020       14-05-22      -12.0           RT2     makeyourtrip
```

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

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

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

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

df_bookings.shape
```

```
(134578, 12)
```

(2) Outlier removal in revenue generated

```
df_bookings.revenue_generated.min(),
df_bookings.revenue_generated.max()
```

```
(6500, 28560000)
```

```
df_bookings.revenue_generated.mean(),
df_bookings.revenue_generated.median()
```

```
(15378.036937686695, 13500.0)
```

```
avg, std = df_bookings.revenue_generated.mean(),
df_bookings.revenue_generated.std()

higher_limit = avg + 3*std
higher_limit
```

```
294498.50173207896

lower_limit = avg - 3*std
lower_limit

-263742.4278567056

df_bookings[df_bookings.revenue_generated<=0]

Empty DataFrame
Columns: [booking_id, property_id, booking_date, check_in_date,
checkout_date, no_guests, room_category, booking_platform,
ratings_given, booking_status, revenue_generated, revenue_realized]
Index: []

df_bookings[df_bookings.revenue_generated>higher_limit]

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

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

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

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

(134573, 12)

df_bookings.revenue_realized.describe()
```

```
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
```

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

```
33479.358661845814
```

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

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

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

```
4.0
134474      6/8/2022         5.0           RT4   direct offline
5.0
134581      1/8/2022         4.0           RT4     makeyourtrip
4.0

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

[1299 rows x 12 columns]
```

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

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

```
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
```

```
# mean + 3*standard deviation
23439+3*9048
```

```
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

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

```
Empty DataFrame
Columns: [booking_id, property_id, booking_date, check_in_date,
checkout_date, no_guests, room_category, booking_platform,
ratings_given, booking_status, revenue_generated, revenue_realized]
Index: []
```

```
df_bookings.isnull().sum()
```

```
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

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)

```
df_agg_bookings.isnull().sum()
```

```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               2
dtype: int64
```

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

```
    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
```

```
df_agg_bookings.capacity.median()
```

```
25.0

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

df_agg_bookings.loc[[8,15]]

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

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

```
df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.ca
pacity]

      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

df_agg_bookings.shape

(9200, 5)

df_agg_bookings =
df_agg_bookings[df_agg_bookings.successful_bookings<=df_agg_bookings.c
apacity]
df_agg_bookings.shape

(9194, 5)
```

==> 3. Data Transformation

Create occupancy percentage column

```
df_agg_bookings.head(3)
```

```
   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
```

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

You can use following approach to get rid of SettingWithCopyWarning

```
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)
```

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

    occ_pct
0  0.833333
1  0.933333
2  0.766667
```

Convert it to a percentage value

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

```
   property_id check_in_date room_category  successful_bookings
capacity  \
0        16559      1-May-22           RT1                   25
30.0
```

```
1        19562      1-May-22           RT1                    28
30.0
2        19563      1-May-22           RT1                    23
30.0

   occ_pct
0    83.33
1    93.33
2    76.67
```

```
df_bookings.head()
```

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

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

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

```
df_agg_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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?

```
df_agg_bookings.head(3)
```

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

   occ_pct
0    83.33
1    93.33
2    76.67
```

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

```
room_category
RT1    57.889643
RT2    58.009756
RT3    58.028213
RT4    59.277925
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

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

```
   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        16558      1-May-22           RT1                   18
19.0

   occ_pct room_id room_class
0    83.33     RT1   Standard
1    93.33     RT1   Standard
2    76.67     RT1   Standard
3    94.74     RT1   Standard
```

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

```
   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        16558      1-May-22           RT1                   18
19.0

   occ_pct room_class
0    83.33   Standard
1    93.33   Standard
2    76.67   Standard
3    94.74   Standard
```

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

```
room_class
Elite           58.009756
Premium         58.028213
Presidential    59.277925
```

```
Standard        57.889643
Name: occ_pct, dtype: float64

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

57.88964285714285
```

2. Print average occupancy rate per city

```
df_hotels.head(3)

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

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

   property_id check_in_date room_category  successful_bookings
capacity  \
0        16559      1-May-22           RT1                   25
30.0
1        16559      2-May-22           RT1                   20
30.0
2        16559      3-May-22           RT1                   17
30.0

   occ_pct room_class  property_name category    city
0    83.33   Standard  Atliq Exotica   Luxury  Mumbai
1    66.67   Standard  Atliq Exotica   Luxury  Mumbai
2    56.67   Standard  Atliq Exotica   Luxury  Mumbai

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

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

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

```
df_date.head(3)

        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
```

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

```
   property_id check_in_date room_category  successful_bookings
capacity  \
0        16559     10-May-22           RT1                   18
30.0
1        16559     10-May-22           RT2                   25
41.0
2        16559     10-May-22           RT3                   20
32.0

   occ_pct room_class  property_name category    city       date  mmm
yy  \
0    60.00   Standard  Atliq Exotica   Luxury  Mumbai  10-May-22  May
22
1    60.98      Elite  Atliq Exotica   Luxury  Mumbai  10-May-22  May
22
2    62.50    Premium  Atliq Exotica   Luxury  Mumbai  10-May-22  May
22

  week no  day_type
0    W 20  weekeday
1    W 20  weekeday
2    W 20  weekeday
```

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

```
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

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

```
      property_id check_in_date room_category  successful_bookings
capacity  \
2200        16559     10-Jun-22           RT1                   20
30.0
2201        16559     10-Jun-22           RT2                   26
41.0
2202        16559     10-Jun-22           RT3                   20
32.0
2203        16559     10-Jun-22           RT4                   11
18.0

      occ_pct    room_class  property_name category    city       date
```

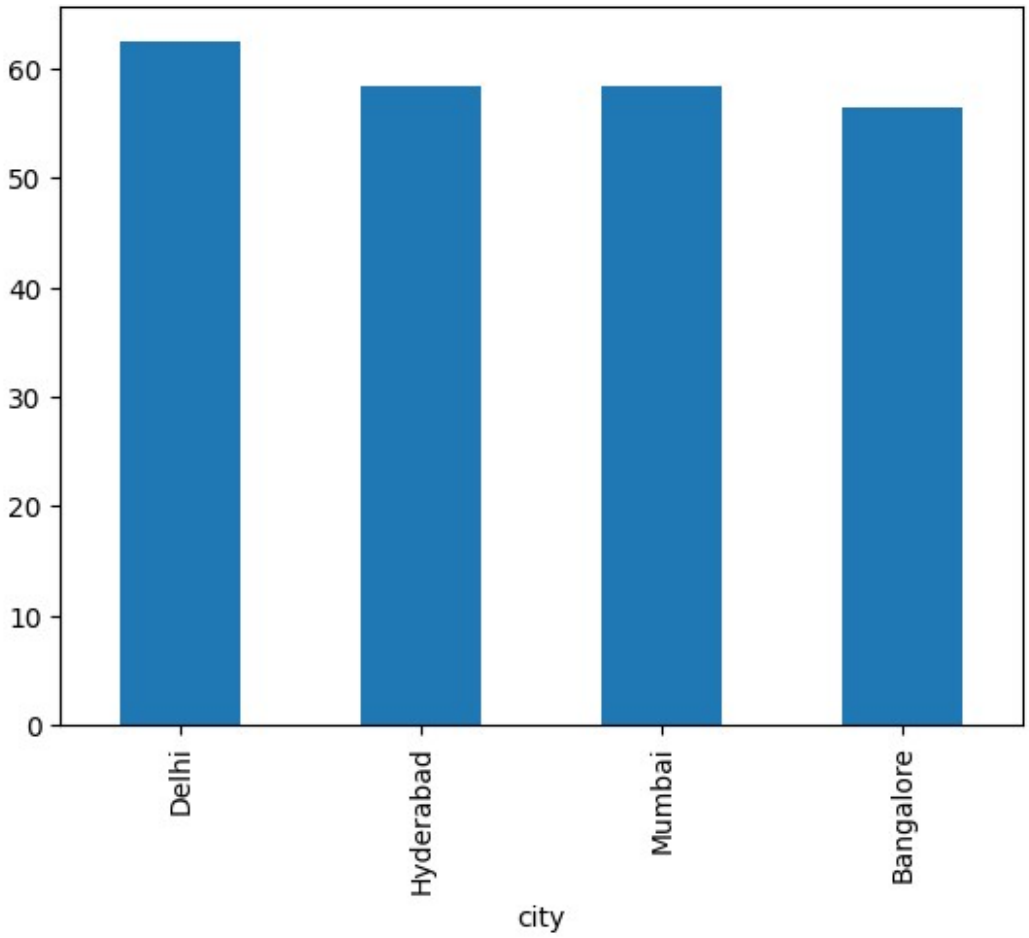
```
\
2200    66.67      Standard  Atliq Exotica   Luxury  Mumbai  10-Jun-22

2201    63.41         Elite  Atliq Exotica   Luxury  Mumbai  10-Jun-22

2202    62.50       Premium  Atliq Exotica   Luxury  Mumbai  10-Jun-22

2203    61.11  Presidential  Atliq Exotica   Luxury  Mumbai  10-Jun-22


      mmm yy week no  day_type
2200  Jun 22    W 24  weekeday
2201  Jun 22    W 24  weekeday
2202  Jun 22    W 24  weekeday
2203  Jun 22    W 24  weekeday
```

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

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

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

```
<AxesSubplot: xlabel='city'>
```



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

```
df_august = pd.read_csv("datasets/new_data_august.csv")
df_august.head(3)
```

```
   property_id  property_name  category       city room_category
room_class  \
0        16559  Atliq Exotica    Luxury     Mumbai           RT1
Standard
1        19562      Atliq Bay    Luxury  Bangalore           RT1
Standard
2        19563   Atliq Palace  Business  Bangalore           RT1
Standard

  check_in_date  mmm yy week no  day_type  successful_bookings
capacity  \
0     01-Aug-22  Aug-22    W 32  weekeday                    30
30
1     01-Aug-22  Aug-22    W 32  weekeday                    21
30
2     01-Aug-22  Aug-22    W 32  weekeday                    23
```

```
30
```

```
     occ%
0  100.00
1   70.00
2   76.67
```

```
df_august.columns
```

```
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')
```

```
df.columns
```

```
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')
```

```
df_august.shape
```

```
(7, 13)
```

```
df.shape
```

```
(6497, 14)
```

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

```
      property_id check_in_date room_category  successful_bookings
capacity  \
6494        16563     31-Jul-22           RT2                   32
38.0
6495        16563     31-Jul-22           RT3                   14
20.0
6496        16563     31-Jul-22           RT4                   13
18.0
6497        16559     01-Aug-22           RT1                   30
30.0
6498        19562     01-Aug-22           RT1                   21
30.0
6499        19563     01-Aug-22           RT1                   23
30.0
6500        19558     01-Aug-22           RT1                   30
40.0
6501        19560     01-Aug-22           RT1                   20
```

```
26.0
6502        17561     01-Aug-22           RT1                    18
26.0
6503        17564     01-Aug-22           RT1                    10
16.0

      occ_pct    room_class  property_name  category       city
date  \
6494    84.21         Elite   Atliq Palace  Business      Delhi  31-
Jul-22
6495    70.00       Premium   Atliq Palace  Business      Delhi  31-
Jul-22
6496    72.22  Presidential   Atliq Palace  Business      Delhi  31-
Jul-22
6497      NaN      Standard  Atliq Exotica    Luxury     Mumbai
NaN
6498      NaN      Standard      Atliq Bay    Luxury  Bangalore
NaN
6499      NaN      Standard   Atliq Palace  Business  Bangalore
NaN
6500      NaN      Standard   Atliq Grands    Luxury  Bangalore
NaN
6501      NaN      Standard     Atliq City  Business  Bangalore
NaN
6502      NaN      Standard      Atliq Blu    Luxury     Mumbai
NaN
6503      NaN      Standard  Atliq Seasons  Business     Mumbai
NaN

      mmm yy week no  day_type    occ%
6494  Jul 22    W 32   weekend     NaN
6495  Jul 22    W 32   weekend     NaN
6496  Jul 22    W 32   weekend     NaN
6497  Aug-22    W 32  weekeday  100.00
6498  Aug-22    W 32  weekeday   70.00
6499  Aug-22    W 32  weekeday   76.67
6500  Aug-22    W 32  weekeday   75.00
6501  Aug-22    W 32  weekeday   76.92
6502  Aug-22    W 32  weekeday   69.23
6503  Aug-22    W 32  weekeday   62.50
```

```
latest_df.shape
```

```
(6504, 15)
```

6. Print revenue realized per city

```
df_bookings.head()
```

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

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

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

df_hotels.head(3)

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

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

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

```
3/5/2022

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

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

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

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

7. Print month by month revenue

```
df_date.head(3)

        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

df_date["mmm yy"].unique()

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

df_bookings_all.head(3)

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

   no_guests room_category booking_platform  ratings_given
booking_status  \
```

```
0        2.0           RT1           others             NaN
Cancelled
1        4.0           RT1    direct online             5.0    Checked
Out
2        2.0           RT1           others             4.0    Checked
Out
```

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

```
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
```

```
df_date["date"] = pd.to_datetime(df_date["date"])
df_date.head(3)
```

```
        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
```

```
df_bookings_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 16.4+ MB
```

```
df_bookings_all["check_in_date"] =
pd.to_datetime(df_bookings_all["check_in_date"])
df_bookings_all.head(4)
```

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

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

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

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

```
         booking_id  property_id booking_date check_in_date
checkout_date  \
0  May052216558RT11        16558     15-04-22    2022-05-05
7/5/2022
1  May052216558RT12        16558     30-04-22    2022-05-05
7/5/2022
```

```
2  May052216558RT13        16558     1/5/2022    2022-05-05
6/5/2022

   no_guests room_category booking_platform  ratings_given
booking_status  \
0        3.0           RT1         tripster            5.0    Checked
Out
1        2.0           RT1           others            NaN
Cancelled
2        3.0           RT1   direct offline            5.0    Checked
Out

   revenue_generated  revenue_realized property_name category
city  \
0              10010             10010  Atliq Grands   Luxury  Delhi

1               9100              3640  Atliq Grands   Luxury  Delhi

2              10010             10010  Atliq Grands   Luxury  Delhi


        date  mmm yy week no  day_type
0 2022-05-05  May 22    W 19  weekeday
1 2022-05-05  May 22    W 19  weekeday
2 2022-05-05  May 22    W 19  weekeday
```

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

```
mmm yy
Jul 22    389940912
Jun 22    377191229
May 22    408375641
Name: revenue_realized, dtype: int64
```

Exercise-1. Print revenue realized per hotel type

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

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

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

```
property_name
Atliq Seasons     66086735
Atliq Grands     211462134
Atliq Bay        259996918
Atliq Blu        260851922
Atliq City       285798439
Atliq Palace     304081863
```

```
Atliq Exotica    320258588
Name: revenue_realized, dtype: int64
```

Exercise-2 Print average rating per city

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

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

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

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

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