AtliQ Hotels Data Analysis Project

In [1]:
```
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

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

Explore bookings data

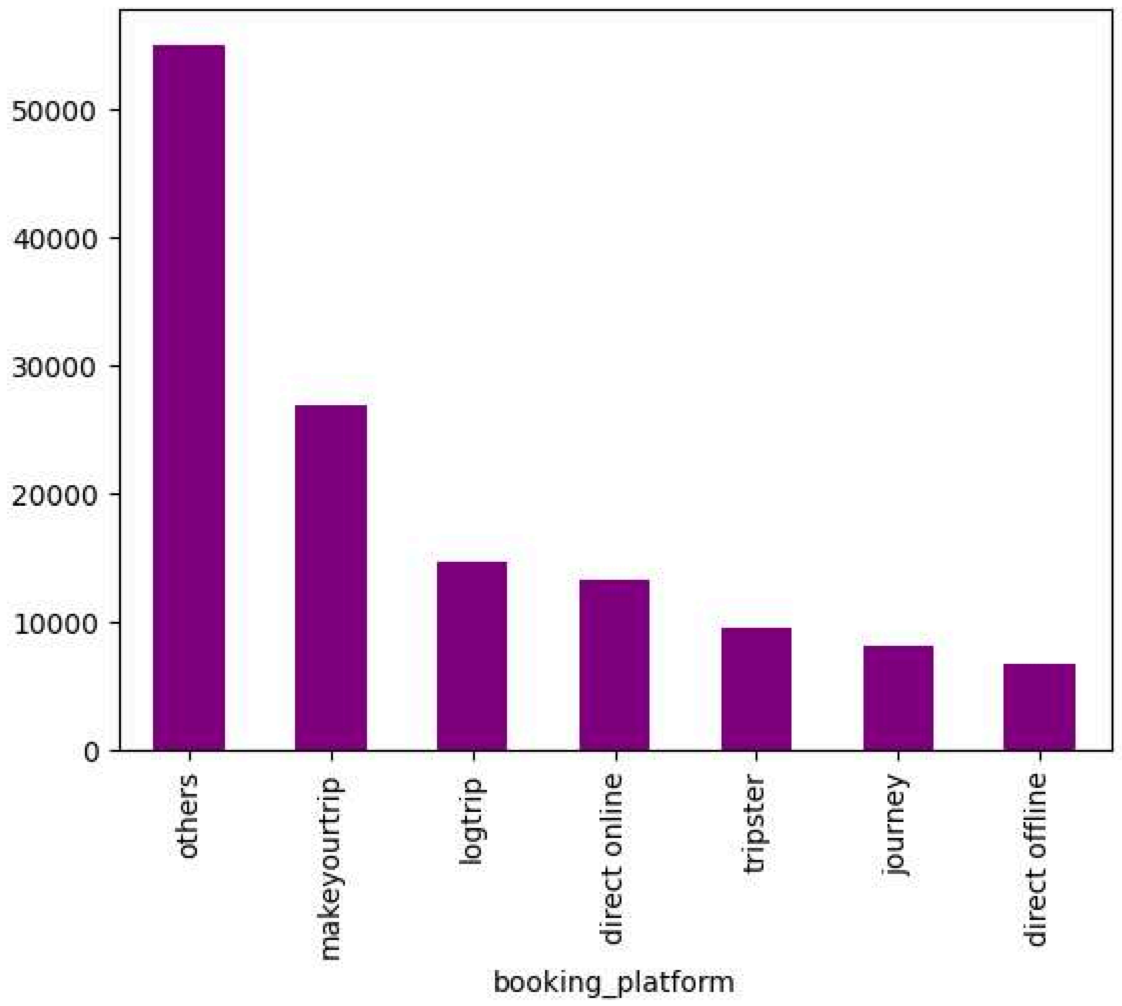
In [3]:
```
df_bookings.head(5)
```

Out[3]:

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

In [248…
```
df_bookings.shape
```
Out[248…
```
(134590, 12)
```
In [249…
```
df_bookings.room_category.unique()
```
Out[249…
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```
In [250…
```
df_bookings.booking_platform.unique()
```
Out[250…
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```
In [251…
```
df_bookings.booking_platform.value_counts()
```
Out[251…
```
others            55066
makeyourtrip      26898
logtrip           14756
direct online     13379
tripster           9630
journey            8106
direct offline     6755
Name: booking_platform, dtype: int64
```
In [4]:
```
df_bookings.booking_platform.value_counts().plot(kind="bar",color="purple")
```

```
Out[4]: <Axes: xlabel='booking_platform'>
```



```
In [253…  df_bookings.describe()
```

Out[253…

	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 [3]:
```
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 [255…
```
df_hotels.shape
```

Out[255…
```
(25, 4)
```

In [256…
```
df_hotels.head(3)
```

Out[256…

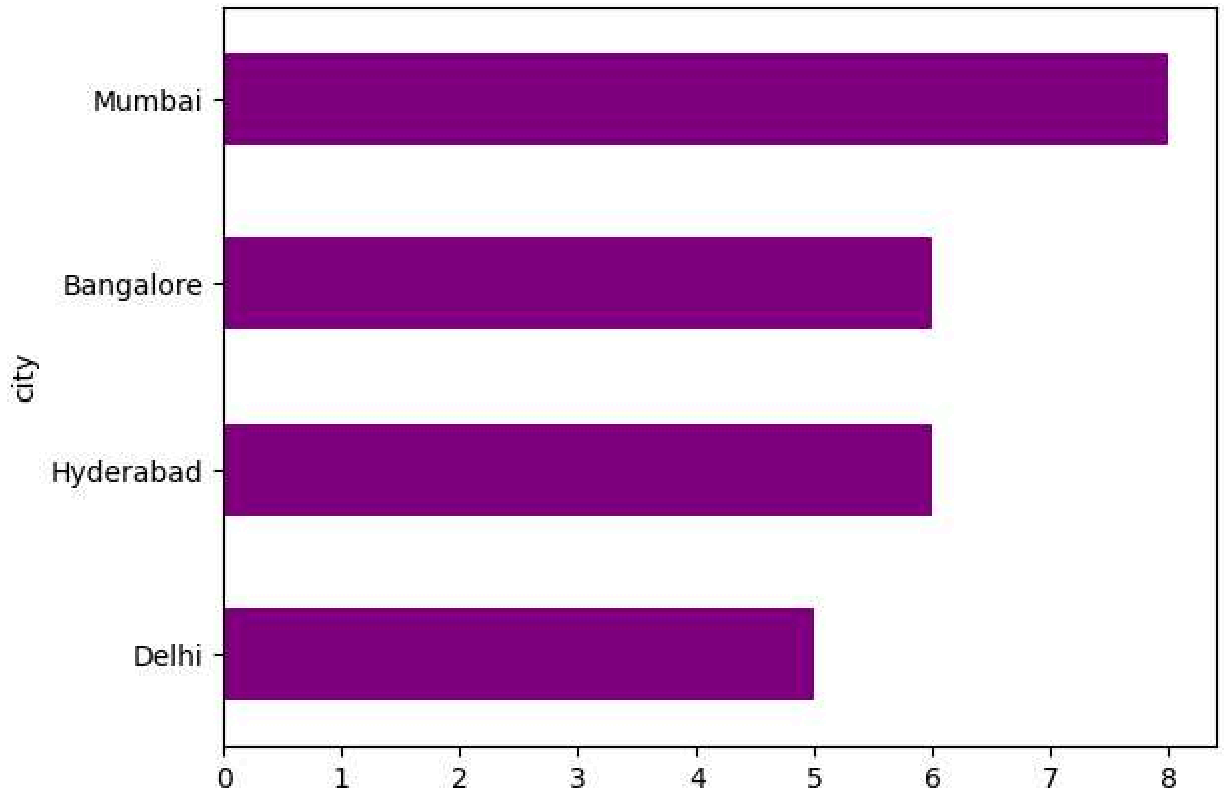
	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi

```
In [257…  df_hotels.category.value_counts()
```

```
Out[257…  Luxury      16
          Business     9
          Name: category, dtype: int64
```

```
In [10]:  df_hotels.city.value_counts().sort_values().plot(kind="barh",color="purple")
```

```
Out[10]:  <Axes: ylabel='city'>
```



Exercise: Explore aggregate bookings

```
In [259…  df_agg_bookings.head(3)
```

Out[259…

	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

```
In [4]:  df_agg_bookings.property_id.unique()
```

```
Out[4]:  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])
```

Exercise-2. Find out total bookings per property_id

```
In [6]:  df_agg_bookings.groupby('property_id')['successful_bookings'].sum()
```

```
Out[6]: 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

```
In [7]: df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.capacity]
```

Out[7]:

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

Exercise-4. Find out properties that have highest capacity

In [8]:
```
df_agg_bookings.capacity.max()
```

Out[8]:
```
np.float64(50.0)
```

==> 2. Data Cleaning

In [265…
```
df_bookings.describe()
```

Out[265...

	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 [4]:
```
df_bookings = df_bookings[df_bookings.no_guests>0]
df_bookings
```

Out[4]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratin
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	
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	
...	...	...	...	...	...	...	...	...	
134584	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	2.0	RT4	others	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	1.0	RT4	makeyourtrip	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	1.0	RT4	tripster	
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	

134578 rows × 12 columns

In [268…
```
df_bookings.shape
```
Out[268…
```
(134578, 12)
```

(2) Outlier removal in revenue generated

In [269…
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```
Out[269…
```
(6500, 28560000)
```

In [270…
```
df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median()
```
Out[270…
```
(15378.036937686695, 13500.0)
```

In [5]:
```
avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.std()
```

In [6]:
```
higher_limit = avg + 3*std
higher_limit
```

Out[6]:
```
np.float64(294498.50173207896)
```

In [273…
```
lower_limit = avg - 3*std
lower_limit
```

Out[273…
```
-263742.4278567056
```

In [274…
```
df_bookings[df_bookings.revenue_generated<=0]
```

Out[274…

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	book

In [7]:
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

Out[7]:

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

In [7]:
```
df_bookings = df_bookings[df_bookings.revenue_generated<=higher_limit]
df_bookings.shape
```

Out[7]:
```
(134573, 12)
```

```
In [277…  df_bookings.revenue_realized.describe()
```

```
Out[277…  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 [9]:  higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std()
         higher_limit
```

```
Out[9]:  np.float64(33479.53674501214)
```

```
In [10]:  df_bookings[df_bookings.revenue_realized>higher_limit]
```

Out[10]:

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

1299 rows × 12 columns

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

In [280…

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

```
Out[280…  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 [281…  # mean + 3*standard deviation
          23439+3*9048
```

```
Out[281…  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 [11]:  df_bookings.isnull().sum()
```

```
Out[11]:  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

Exercise-1. In aggregate bookings find columns that have null values. Fill these null values with whatever you think is the appropriate subtitute (possible ways is to use mean or median)

In [12]:
```
df_agg_bookings.isnull().sum()
```

Out[12]:
```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               2
dtype: int64
```

In [13]:
```
df_agg_bookings[df_agg_bookings.capacity.isna()]
```

Out[13]:

	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 [14]:
```
df_agg_bookings.capacity.median()
```

Out[14]:
```
np.float64(25.0)
```

In [15]:
```
df_agg_bookings.capacity.fillna(df_agg_bookings.capacity.median(),inplace=True)
df_agg_bookings.loc[[8,14]]
```

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_5036\4225626067.py:1: FutureWarning: A value is trying to be set on a copy of a Da
taFrame 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 intermediate object on which we are set
ting values always behaves as a copy.

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


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

Out[15]:

	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

==> 3. Data Transformation

Create occupancy percentage column

In [10]:
```
df_agg_bookings.head(3)
```

Out[10]:

	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 [293…
```
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

In [11]:
```
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[11]:

	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 [12]:
```
df_agg_bookings['occ_pct'] = df_agg_bookings['occ_pct'].apply(lambda x: round(x*100, 2))
df_agg_bookings.head(3)
```

Out[12]:

	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 [299…
```
df_bookings.head()
```

Out[299…

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

In [297…
```
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?

In [35]:
```
df_agg_bookings.head(3)
```

Out[35]:

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

```
Out[36]: room_category
         RT1    58.232748
         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 [13]: df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_on="room_id")
         df.head(4)
```

Out[13]:

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

Out[14]:

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

Out[39]:
```
room_class
Elite           58.040278
Premium         58.028213
Presidential    59.300461
Standard        58.232748
Name: occ_pct, dtype: float64
```

In [40]:
```
df[df.room_class=="Standard"].occ_pct.mean()
```

Out[40]:
```
np.float64(58.23274782608696)
```

2. Print average occupancy rate per city

In [310…
```
df_hotels.head(3)
```

Out[310…

	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 [15]:
```
df = pd.merge(df, df_hotels, on="property_id")
df.head(3)
```

Out[15]:

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

In [18]:
```
df.groupby("city_x")["occ_pct"].mean()
```

Out[18]:

```
city_x
Bangalore    56.594207
Delhi        61.606467
Hyderabad    58.144651
Mumbai       57.943142
Name: occ_pct, dtype: float64
```

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

In [314…

```
df_date.head(3)
```

Out[314…

	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 [17]:

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

Out[17]:

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

In [21]:

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

```
Out[21]: 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

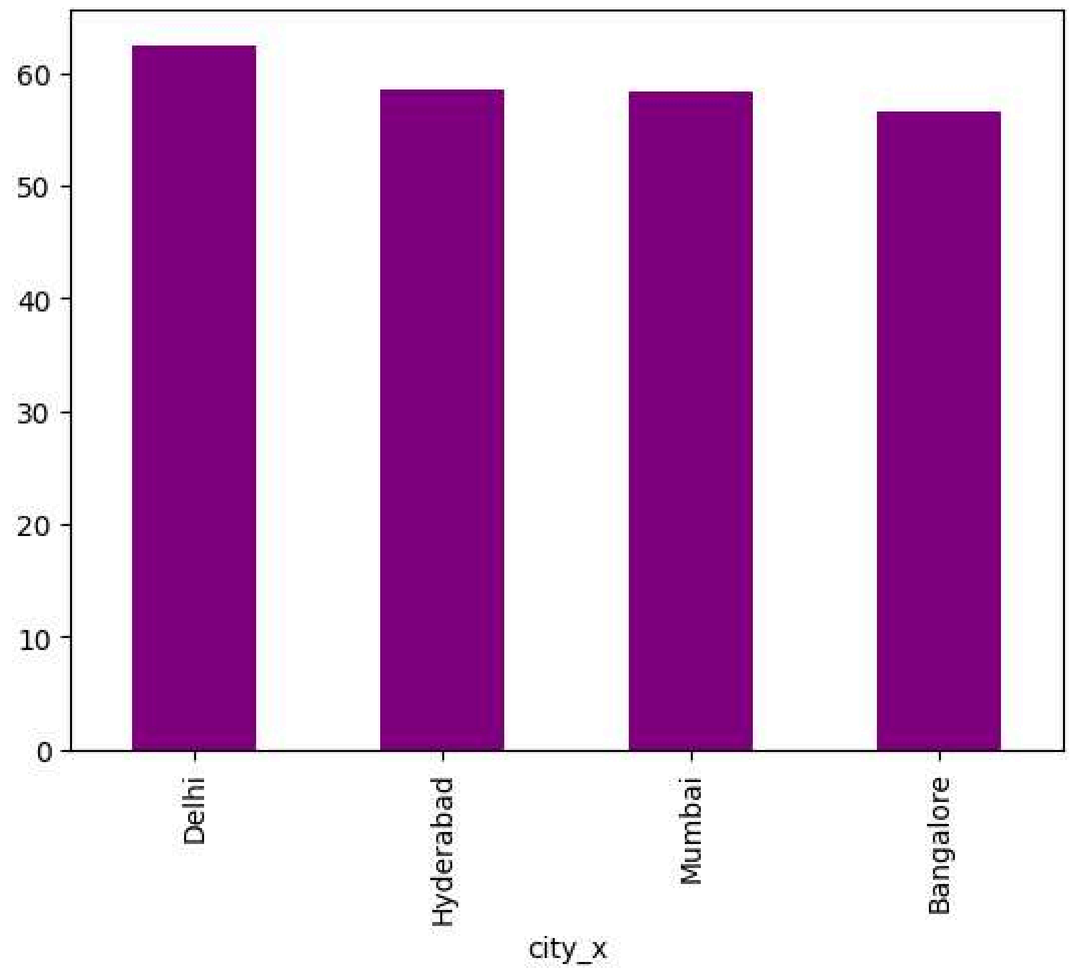
```
In [18]: df_june_22 = df[df["mmm yy"]=="Jun 22"]
         df_june_22.head(4)
```

Out[18]:

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

```
In [30]: df_june_22.groupby('city_x')['occ_pct'].mean().round(2).sort_values(ascending=False).plot(kind="bar",color="purple")
```

```
Out[30]: <Axes: xlabel='city_x'>
```



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

```
In [16]: df_august = pd.read_csv("datasets/new_data_august.csv")
         df_august.head(3)
```

Out[16]:

	property_id	property_name	category	city	room_category	room_class	check_in_date	mmm yy	week no	day_type	successful_bookir
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 [334…
```
df_august.columns
```

Out[334…
```
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 [332…
```
df.columns
```

Out[332…
```
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 [337…
```
df_august.shape
```

Out[337…
```
(7, 13)
```

In [338…
```
df.shape
```

Out[338…
```
(6497, 14)
```

In [19]:
```
latest_df = pd.concat([df, df_august], ignore_index = True, axis = 0)
latest_df.tail(6)
```

Out[19]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city
6501	19562	01-Aug-22	RT1	21	30.0	NaN	Standard	Atliq Bay	Luxury	Bangalore
6502	19563	01-Aug-22	RT1	23	30.0	NaN	Standard	Atliq Palace	Business	Bangalore
6503	19558	01-Aug-22	RT1	30	40.0	NaN	Standard	Atliq Grands	Luxury	Bangalore
6504	19560	01-Aug-22	RT1	20	26.0	NaN	Standard	Atliq City	Business	Bangalore
6505	17561	01-Aug-22	RT1	18	26.0	NaN	Standard	Atliq Blu	Luxury	Mumbai
6506	17564	01-Aug-22	RT1	10	16.0	NaN	Standard	Atliq Seasons	Business	Mumbai

In [339…

```
latest_df.shape
```

Out[339…

```
(6504, 15)
```

6. Print revenue realized per city

In [40]:

```
df_agg_bookings.head()
```

Out[40]:

	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
3	17558	1-May-22	RT1	30	19.0	157.89
4	16558	1-May-22	RT1	18	19.0	94.74

In [ ]:

```
df_all = pd.merge(df_agg_bookings,df_bookings,on = "property_id")
df_all.head(3)
```

In [345…

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

Out[345…

	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 [21]:

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

Out[21]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_giv
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	Na
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4

In [361…

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

Out[361…

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

7. Print month by month revenue

In [356…

```
df_date.head(3)
```

Out[356…

	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 [357…

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

Out[357…

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

In [363…

```
df_bookings_all.head(3)
```

Out[363…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_give
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	Na
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4

In [364…

```
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 [36]:
```python
df_bookings_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134573 entries, 0 to 134572
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   booking_id         134573 non-null  object 
 1   property_id        134573 non-null  int64  
 2   booking_date       134573 non-null  object 
 3   check_in_date      134573 non-null  object 
 4   checkout_date      134573 non-null  object 
 5   no_guests          134573 non-null  float64
 6   room_category      134573 non-null  object 
 7   booking_platform   134573 non-null  object 
 8   ratings_given      56676 non-null   float64
 9   booking_status     134573 non-null  object 
 10  revenue_generated  134573 non-null  int64  
 11  revenue_realized   134573 non-null  int64  
 12  property_name      134573 non-null  object 
 13  category           134573 non-null  object 
 14  city               134573 non-null  object 
dtypes: float64(2), int64(3), object(10)
memory usage: 15.4+ MB
```

In [8]:
```python
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on="check_in_date", right_on="date")
df_bookings_all.head(3)
```

Out[8]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	book

In [375…

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

Out[375…

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

Exercise-1. Print revenue realized per hotel type

In [22]:

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

Out[22]:

```
category
Business     655967037
Luxury      1052569562
Name: revenue_realized, dtype: int64
```

Exercise-2 Print average rating per city

In [27]:

```
df_bookings_all.groupby("city")["ratings_given"].mean().sort_values(ascending=False)
```

Out[27]:

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

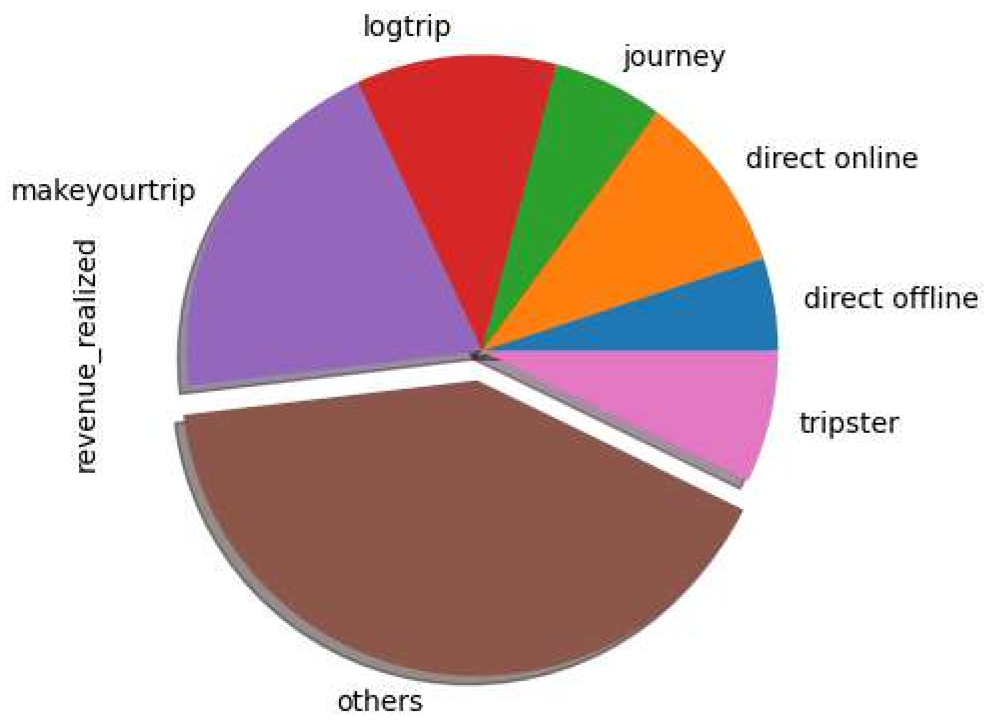
Exercise-3 Print a pie chart of revenue realized per booking platform

In [42]:

```
df_bookings_all.groupby("booking_platform")["revenue_realized"].sum().plot(kind="pie",explode=(0,0,0,0,0,0.1,0),shadow=True)
```

Out[42]:

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


logtrip
journey
direct online
makeyourtrip
direct offline
revenue_realized
tripster
others