Hotel_analysis_project

June 27, 2025

AtliQ Hotels Data Analysis Project

[90]:
```
import pandas as pd
```

0.0.1 ==> 1. Data Import and Data Exploration

0.0.2 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

[91]:
```
df_bookings = pd.read_csv('datasets/fact_bookings.csv')
```

Explore bookings data

[92]:
```
df_bookings.head()
```

[92]:

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

```
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
```

[93]:
```
df_bookings.shape
```

[93]:
```
(134590, 12)
```

[94]:
```
df_bookings.room_category.unique()
```

[94]:
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

[95]:
```
df_bookings.booking_platform.unique()
```

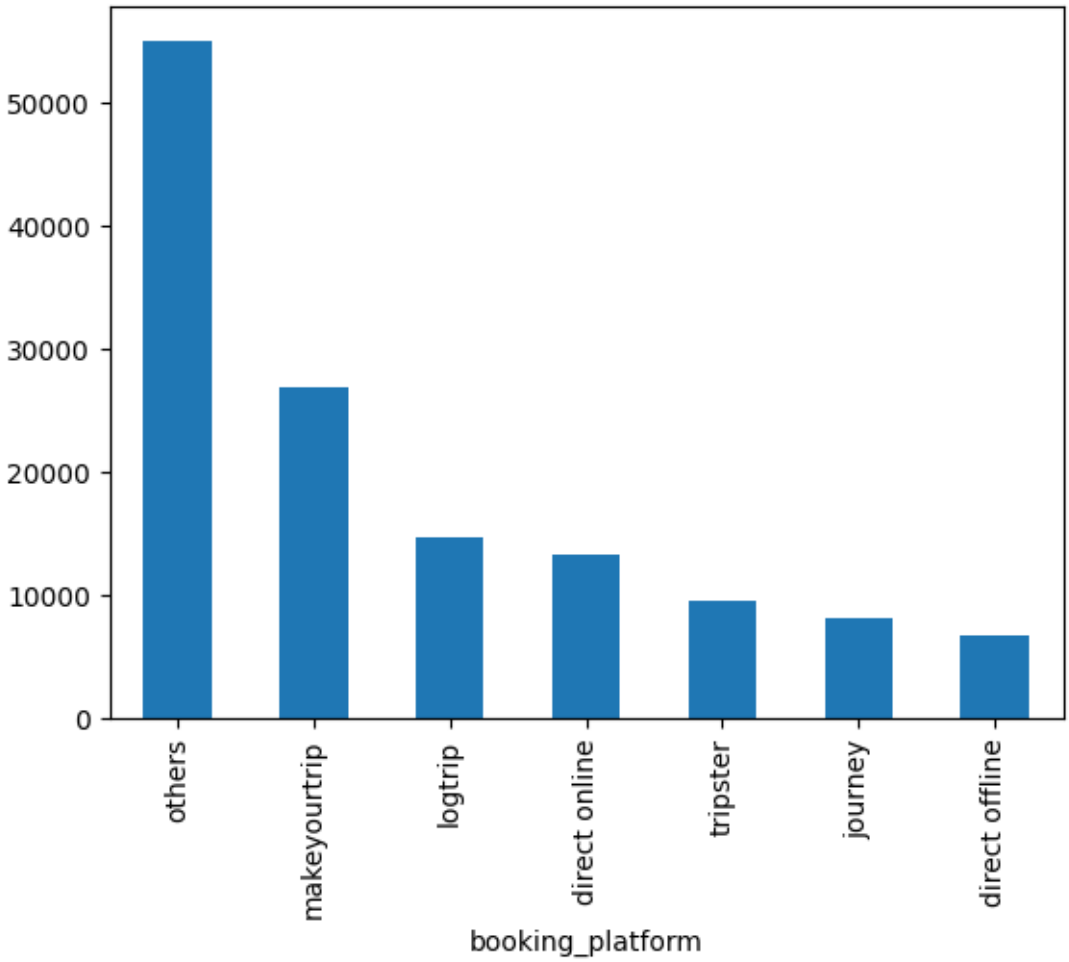
[95]:
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

[96]:
```
df_bookings.booking_platform.value_counts()
```

[96]:
```
booking_platform
others            55066
makeyourtrip      26898
logtrip           14756
direct online     13379
tripster           9630
journey            8106
direct offline     6755
Name: count, dtype: int64
```

[97]:
```
df_bookings.booking_platform.value_counts().plot(kind="bar")
```

[97]:
```
<Axes: xlabel='booking_platform'>
```



[98]:
```
df_bookings.describe()
```

[98]:

	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	
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	

```
25%         7600.000000
50%        11700.000000
75%        15300.000000
max        45220.000000
```

Read rest of the files

[99]:
```
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')
```

[100]:
```
df_hotels.shape
```

[100]:
```
(25, 4)
```

[101]:
```
df_hotels.head(3)
```

[101]:

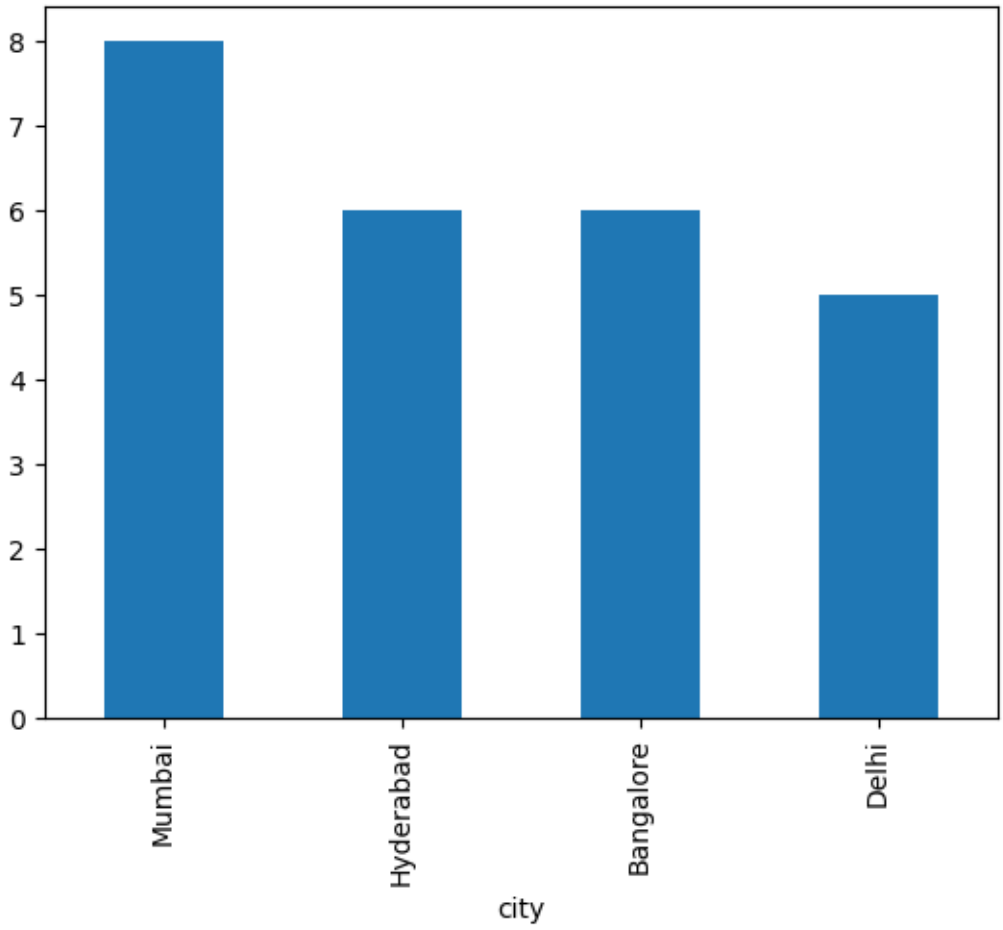
	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi

[102]:
```
df_hotels.category.value_counts()
```

[102]:
```
category
Luxury      16
Business     9
Name: count, dtype: int64
```

[103]:
```
df_hotels.city.value_counts().plot(kind="bar")
```

[103]:
```
<Axes: xlabel='city'>
```



Exercise: Explore aggregate bookings ***

[104]:
```
df_agg_bookings.head(3)
```

[104]:

	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

[105]:
```
df_agg_bookings.property_id.unique()
```

[105]:
```
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

[106]:
```
df_agg_bookings.groupby("property_id")["successful_bookings"].sum()
```

[106]:
```
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

[107]:
```
df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.capacity]
```

[107]:

	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

[108]:
```
df_agg_bookings.capacity.max()
```

[108]:
```
np.float64(50.0)
```

[109]:
```
df_agg_bookings[df_agg_bookings.capacity==df_agg_bookings.capacity.max()]
```

[109]:

	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]

0.0.3 ==> 2. Data Cleaning

---

[110]:
```
df_bookings.describe()
```

[110]:

	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

[111]:
```
df_bookings[df_bookings.no_guests<=0]
```

[111]:

	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.

[112]:
```
df_bookings = df_bookings[df_bookings.no_guests>0]
```

[113]:
```
df_bookings.shape
```

[113]:
```
(134578, 12)
```

(2) Outlier removal in revenue generated

[114]:
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

[114]:
```
(np.int64(6500), np.int64(28560000))
```

[115]:
```
df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median()
```

[115]:
```
(np.float64(15378.036937686695), np.float64(13500.0))
```

[116]:
```
avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.
↪std()
```

[117]:
```
higher_limit = avg + 3*std
higher_limit
```

[117]:
```
np.float64(294498.50173207896)
```

[118]:
```
lower_limit = avg - 3*std
lower_limit
```

[118]:
```
np.float64(-263742.4278567056)
```

[119]:
```
df_bookings[df_bookings.revenue_generated<=0]
```

[119]:
```
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: []
```

[120]:
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

[120]:

	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
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	5.0	Checked Out	9100000	9100
111	May012216559RT32	16559	29-04-22	1/5/2022	2/5/2022	6.0	RT3	direct online	NaN	Checked Out	28560000	28560
315	May012216562RT22	16562	28-04-22	1/5/2022	4/5/2022	2.0	RT2	direct offline	3.0	Checked Out	12600000	12600
562	May012217559RT118	17559	26-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancelled	2000000	4420
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	2.0	RT2	direct online	3.0	Checked Out	10000000	12600

[121]:
```
df_bookings = df_bookings[df_bookings.revenue_generated<=higher_limit]
df_bookings.shape
```

[121]: (134573, 12)

[122]:
```
df_bookings.revenue_realized.describe()
```

[122]:
```
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
```

[123]:
```
higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.
 ↪revenue_realized.std()
higher_limit
```

[123]:
```
np.float64(33479.358661845814)
```

[124]:
```
df_bookings[df_bookings.revenue_realized>higher_limit]
```

[124]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	4.0	RT4	others	NaN
139	May012216559RT43	16559	1/5/2022	1/5/2022	2/5/2022	6.0	RT4	tripster	3.0
143	May012216559RT47	16559	28-04-22	1/5/2022	3/5/2022	3.0	RT4	others	5.0
149	May012216559RT413	16559	24-04-22	1/5/2022	7/5/2022	5.0	RT4	logtrip	NaN
222	May012216560RT45	16560	30-04-22	1/5/2022	3/5/2022	5.0	RT4	others	3.0
…	…	…	…	…	…	…	…	…	…
134328	Jul312219560RT49	19560	31-07-22	31-07-22	2/8/2022	6.0	RT4	direct online	5.0
134331	Jul312219560RT412	19560	31-07-22	31-07-22	1/8/2022	6.0	RT4	others	2.0
134467	Jul312219562RT45	19562	28-07-22	31-07-22	1/8/2022	6.0	RT4	makeyourtrip	4.0
134474	Jul312219562RT412	19562	25-07-22	31-07-22	6/8/2022	5.0	RT4	direct offline	5.0
134581	Jul312217564RT42	17564	31-07-22	31-07-22	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

[125]:
```
df_bookings[df_bookings.room_category=="RT4"].revenue_realized.describe()
```

[125]:
```
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
```

[126]:
```
# mean + 3*standard deviation
23439+3*9048
```

[126]:
```
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

[127]:
```
df_bookings[df_bookings.booking_id=="May012216558RT213"]
```

[127]:
```
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: []
```

[128]:
```
df_bookings.isnull().sum()
```

[128]:
```
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)

[129]:
```
df_agg_bookings.isnull().sum()
```

[129]:
```
property_id           0
check_in_date         0
room_category         0
successful_bookings   0
capacity              2
dtype: int64
```

[130]:
```
df_agg_bookings[df_agg_bookings.capacity.isna()]
```

[130]:

	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

[131]:
```
df_agg_bookings.capacity.median()
```

[131]:
```
np.float64(25.0)
```

[132]:
```
df_agg_bookings.capacity.fillna(df_agg_bookings.capacity.median(), inplace=True)
```

```
C:\Users\rudra\AppData\Local\Temp\ipykernel_3980\625765049.py:1: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
```

```
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[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)
```

[133]:
```
df_agg_bookings.loc[[8,15]]
```

[133]:

	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

[134]:
```
df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.capacity]
```

[134]:

	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

[135]:
```
df_agg_bookings.shape
```

[135]:
```
(9200, 5)
```

[136]:
```
df_agg_bookings = df_agg_bookings[df_agg_bookings.
 ↪successful_bookings<=df_agg_bookings.capacity]
df_agg_bookings.shape
```

[136]:
```
(9194, 5)
```

[ ]:
```

```

0.0.4 ==> 3. Data Transformation

Create occupancy percentage column

[137]:
```
df_agg_bookings.head(3)
```

[137]:
```
   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
```

[138]:
```
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

[139]:
```
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)
```

[139]:
```
   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

[140]:
```
df_agg_bookings['occ_pct'] = df_agg_bookings['occ_pct'].apply(lambda x:␣
↪round(x*100, 2))
df_agg_bookings.head(3)
```

[140]:
```
   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
```

[141]:
```
df_bookings.head()
```

[141]:
```
          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
```

[142]:
```
df_agg_bookings.info()
```

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

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

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

0.0.5 ==> 4. Insights Generation

---

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

[143]:
```
df_agg_bookings.head(3)
```

[143]:
```
   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
```

[144]:
```
df_agg_bookings.groupby("room_category")["occ_pct"].mean()
```

[144]:
```
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

[145]:
```
df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category",␣
↪right_on="room_id")
df.head(4)
```

[145]:
```
   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
```

[146]:
```
df.drop("room_id",axis=1, inplace=True)
df.head(4)
```

[146]:
```
   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
```

[147]:
```
df.groupby("room_class")["occ_pct"].mean()
```

[147]:
```
room_class
Elite           58.009756
Premium         58.028213
Presidential    59.277925
Standard        57.889643
Name: occ_pct, dtype: float64
```

[148]:
```
df[df.room_class=="Standard"].occ_pct.mean()
```

[148]:
```
np.float64(57.88964285714285)
```

2. Print average occupancy rate per city

[149]:
```
df_hotels.head(3)
```

[149]:
```
   property_id  property_name  category    city
0        16558   Atliq Grands    Luxury   Delhi
1        16559  Atliq Exotica    Luxury  Mumbai
2        16560     Atliq City  Business   Delhi
```

[150]:
```
df = pd.merge(df, df_hotels, on="property_id")
df.head(3)
```

[150]:
```
   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 room_class  property_name  category       city
0    83.33   Standard  Atliq Exotica    Luxury     Mumbai
1    93.33   Standard      Atliq Bay    Luxury  Bangalore
2    76.67   Standard   Atliq Palace  Business  Bangalore
```

[151]:
```
df.groupby("city")["occ_pct"].mean()
```

[151]:
```
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?

[152]:
```
df_date.head(3)
```

[152]:
```
         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
```

[153]:
```
df = pd.merge(df, df_date, left_on="check_in_date", right_on="date")
df.head(3)
```

[153]:
```
   property_id check_in_date room_category  successful_bookings  capacity  \
0        19563     10-May-22           RT3                   15      29.0
1        18560     10-May-22           RT1                   19      30.0
2        19562     10-May-22           RT1                   18      30.0

   occ_pct room_class property_name  category       city       date  mmm yy  \
0    51.72    Premium  Atliq Palace  Business  Bangalore  10-May-22  May 22
1    63.33   Standard    Atliq City  Business  Hyderabad  10-May-22  May 22
2    60.00   Standard     Atliq Bay    Luxury  Bangalore  10-May-22  May 22

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

[154]:
```
df.groupby("day_type")["occ_pct"].mean().round(2)
```

[154]:
```
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

[155]:
```
df_june_22 = df[df["mmm yy"]=="Jun 22"]
df_june_22.head(4)
```

[155]:
```
      property_id check_in_date room_category  successful_bookings  capacity  \
2200        16559     10-Jun-22           RT1                   20      30.0
2201        19562     10-Jun-22           RT1                   19      30.0
2202        19563     10-Jun-22           RT1                   17      30.0
2203        17558     10-Jun-22           RT1                    9      19.0

      occ_pct room_class  property_name  category       city       date  \
```

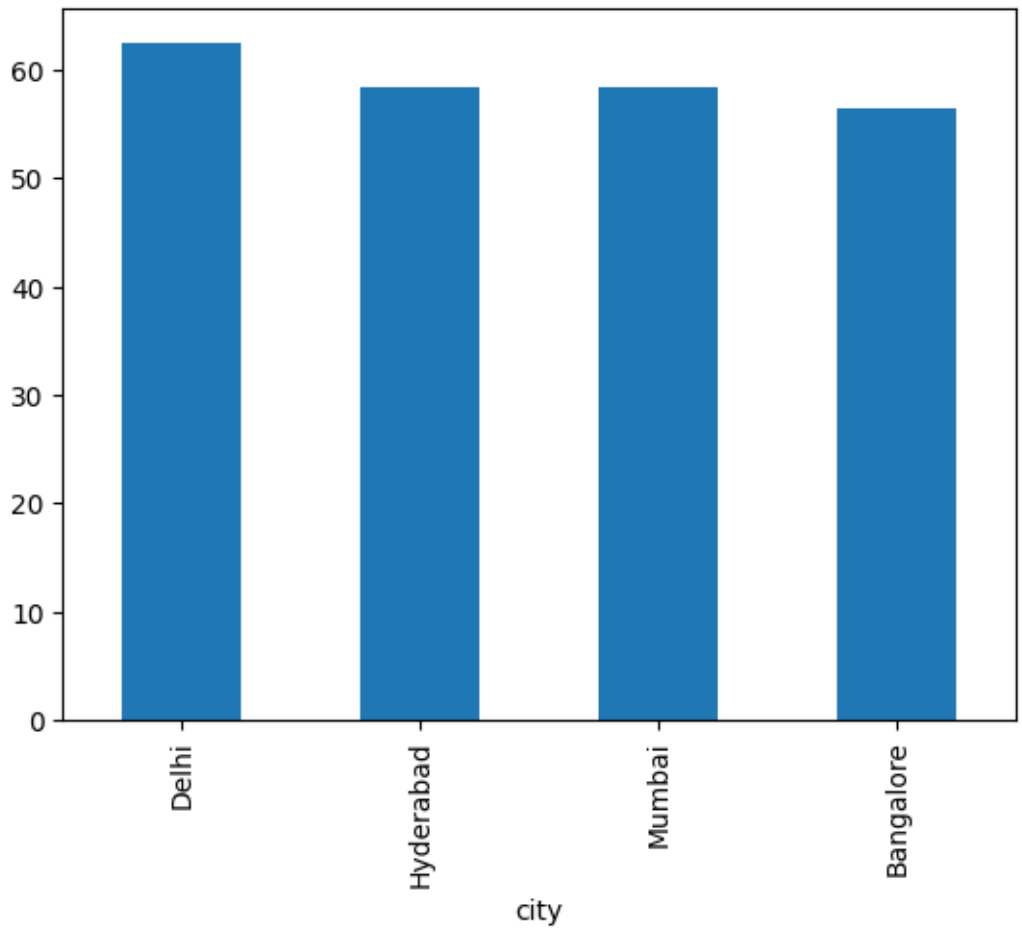
```
2200    66.67   Standard  Atliq Exotica    Luxury     Mumbai  10-Jun-22
2201    63.33   Standard      Atliq Bay    Luxury  Bangalore  10-Jun-22
2202    56.67   Standard   Atliq Palace  Business  Bangalore  10-Jun-22
2203    47.37   Standard   Atliq Grands    Luxury     Mumbai  10-Jun-22

     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
```

[156]:
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).
 ↪sort_values(ascending=False)
```

[156]:
```
city
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Bangalore    56.44
Name: occ_pct, dtype: float64
```

[157]:
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).
 ↪sort_values(ascending=False).plot(kind="bar")
```

[157]:
```
<Axes: xlabel='city'>
```



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

[158]:
```
df_august = pd.read_csv("datasets/new_data_august.csv")
df_august.head(3)
```

[158]:

	property_id	property_name	category	city	room_category	room_class	check_in_date	mmm yy	week no	day_type	successful_bookings	capacity	occ%
0	16559	Atliq Exotica	Luxury	Mumbai	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	30	30	100.00
1	19562	Atliq Bay	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	21	30	70.00
2	19563	Atliq Palace	Business	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	23	30	

```
2   76.67
```

[159]:
```
df_august.columns
```

[159]:
```
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')
```

[160]:
```
df.columns
```

[160]:
```
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')
```

[161]:
```
df_august.shape
```

[161]:
```
(7, 13)
```

[162]:
```
df.shape
```

[162]:
```
(6497, 14)
```

[163]:
```
latest_df = pd.concat([df, df_august], ignore_index = True, axis = 0)
latest_df.tail(10)
```

[163]:

	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
6494	17558	31-Jul-22	RT4	3	6.0	50.0	Presidential	Atliq Grands	Luxury	Mumbai	31-Jul-22			
6495	19563	31-Jul-22	RT4	3	6.0	50.0	Presidential	Atliq Palace	Business	Bangalore	31-Jul-22			
6496	17561	31-Jul-22	RT4	3	4.0	75.0	Presidential	Atliq Blu	Luxury	Mumbai	31-Jul-22			
6497	16559	01-Aug-22	RT1	30	30.0	NaN	Standard	Atliq Exotica	Luxury	Mumbai	NaN			
6498	19562	01-Aug-22	RT1	21	30.0	NaN	Standard	Atliq Bay	Luxury	Bangalore	NaN			
6499	19563	01-Aug-22	RT1	23	30.0	NaN	Standard	Atliq Palace	Business	Bangalore	NaN			
6500	19558	01-Aug-22	RT1	30	40.0	NaN	Standard	Atliq Grands	Luxury	Bangalore	NaN			
6501	19560	01-Aug-22	RT1	20	26.0	NaN	Standard	Atliq City	Business	Bangalore	NaN			
6502	17561	01-Aug-22	RT1	18	26.0									
6503	17564	01-Aug-22	RT1	10	16.0									

```
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
```

[164]:
```
latest_df.shape
```

[164]:
```
(6504, 15)
```

Check this post for codebasics resume project challange winner entry: https://www.linkedin.com/posts/ashishbabaria_codebasicsresumeprojectchallenge-data-powerbi-activity-6977940034414886914-dmoJ?utm_source=share&utm_medium=member_desktop

6. Print revenue realized per city

[165]:
```
df_bookings.head()
```

[165]:
```
         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
```

[166]:
```
df_hotels.head(3)
```

[166]:
```
   property_id  property_name  category    city
0        16558   Atliq Grands    Luxury   Delhi
1        16559  Atliq Exotica    Luxury  Mumbai
2        16560     Atliq City  Business   Delhi
```

[167]:
```
df_bookings_all = pd.merge(df_bookings, df_hotels, on="property_id")
df_bookings_all.head(3)
```

[167]:
```
          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
```

[168]:
```
df_bookings_all.groupby("city")["revenue_realized"].sum()
```

[168]:
```
city
Bangalore    420383550
Delhi        294404488
Hyderabad    325179310
Mumbai       668569251
Name: revenue_realized, dtype: int64
```

7. Print month by month revenue

[169]:
```
df_date.head(3)
```

[169]:
```
         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
```

[170]:
```
df_date["mmm yy"].unique()
```

[170]:
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

[171]:
```
df_bookings_all.head(3)
```

[171]:
```
          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
```

[172]:
```
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
```

[173]:
```
df_date["date"] = pd.to_datetime(df_date["date"])
df_date.head(3)
```

```
C:\Users\rudra\AppData\Local\Temp\ipykernel_3980\173964601.py:1: UserWarning:
Could not infer format, so each element will be parsed individually, falling
back to `dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df_date["date"] = pd.to_datetime(df_date["date"])
```

[173]:
```
        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
```

[174]:
```
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
```

[176]:
```
df_bookings_all["check_in_date"] = pd.
  ↪to_datetime(df_bookings_all["check_in_date"], errors='coerce')
df_bookings_all.head(4)
```

[176]:

	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)
```

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

Exercise-1. Print revenue realized per hotel type

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

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

Exercise-2 Print average rating per city

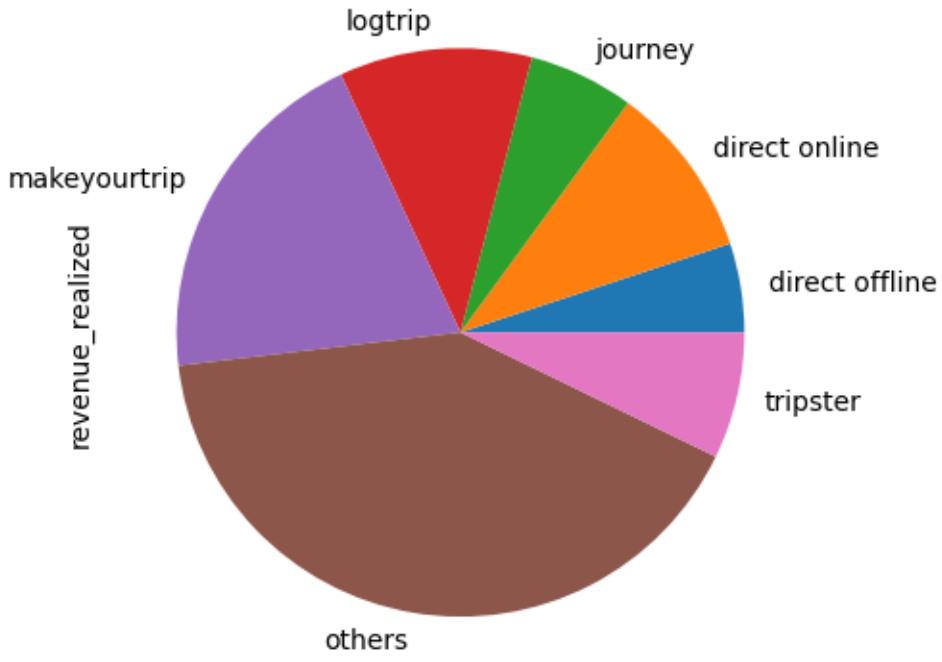
[178]:
```
df_bookings_all.groupby("city")["ratings_given"].mean().round(2)
```

[178]:
```
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

[177]:
```
df_bookings_all.groupby("booking_platform")["revenue_realized"].sum().
↪plot(kind="pie")
```

[177]:
```
<Axes: ylabel='revenue_realized'>
```



[179]:
```

```

```
  Cell In[179], line 1
    jupyter nbconvert Hotel_analysis_project.ipynb --to pdf
            ^
SyntaxError: invalid syntax
```