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	no_guests	room_category	booking_platform	ratings_g
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	

```
df_bookings.shape
```

```
(134590, 12)
```

```
df_bookings.room_category.unique()
```

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

```
df_bookings.booking_platform.unique()
```

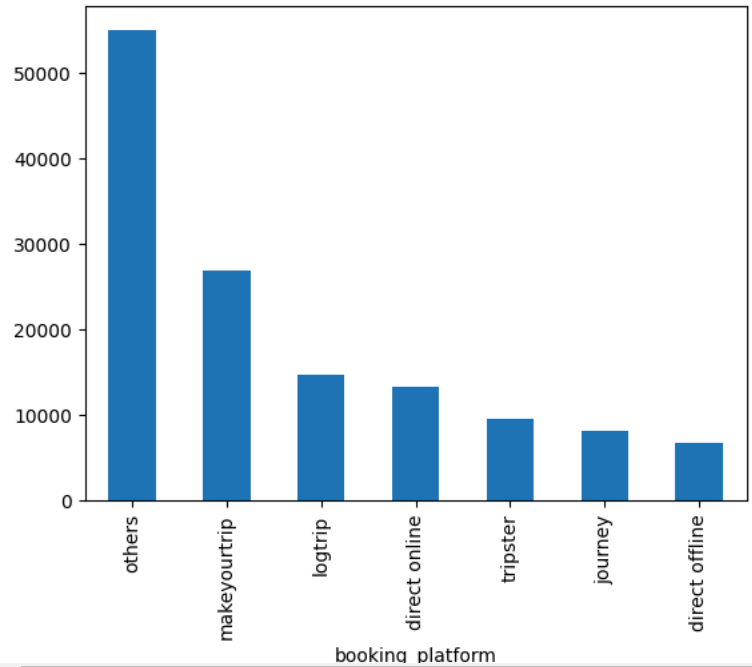
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

```
df_bookings.booking_platform.value_counts()
```

booking_platform	count
others	55066
makeyourtrip	26898
logtrip	14756
direct online	13379
tripster	9630
journey	8106
direct offline	6755

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

```
<Axes: xlabel='booking_platform'>
```




```
df_bookings.describe()
```

	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

```
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

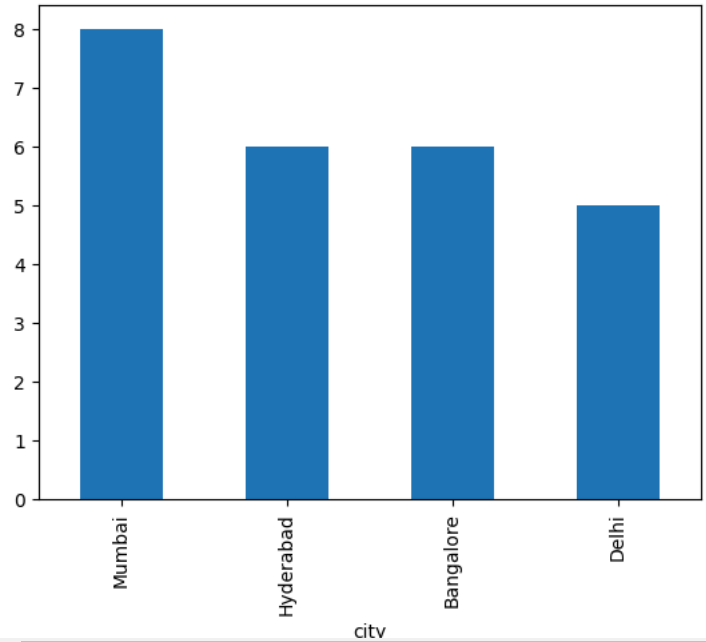
Next steps: Generate code with df_hotels | View recommended plots | New interactive sheet

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

category	count
Luxury	16
Business	9

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

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



Exploring 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

Next steps: Generate code with df_agg_bookings | View recommended plots | New interactive sheet

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])
```

Find out total bookings per property_id

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

property_id	successful_bookings
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

Find out days on which bookings are greater than capacity

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

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

Find out properties that have highest capacity

```
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 × 5 columns

==> 2. Data Cleaning

```
df_bookings.describe()
```

	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

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

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

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
```

```
294498.50173198653
```

```
lower_limit = avg - 3*std
lower_limit
```

```
-263742.4278566132
```

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

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	bo

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

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	rat
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	

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

```
(134573, 12)
```

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

	revenue_realized
count	134573.000000
mean	12695.983585
std	6927.791692
min	2600.000000
25%	7600.000000
50%	11700.000000
75%	15300.000000
max	45220.000000

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

```
33479.3586618449
```

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

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	rat
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

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

	revenue_realized
count	16071.000000
mean	23439.308444
std	9048.599076
min	7600.000000
25%	19000.000000
50%	26600.000000
75%	32300.000000
max	45220.000000

```
# 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"]
```

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	bo

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

	0
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

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

Replacing the null values in aggregate bookings

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

	0
property_id	0
check_in_date	0
room_category	0
successful_bookings	0
capacity	2

```
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)
```

```
<ipython-input-44-84eb537c0f2a>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

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

  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

finding out records in aggregate bookings that have successful_bookings value greater than capacity

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

	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.capacity]
df_agg_bookings.shape
```

```
(9194, 5)
```

```
Start coding or generate with AI.
```

==> 3. Data Transformation

Creating 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

Next steps: Generate code with df_agg_bookings | View recommended plots | New interactive sheet

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

getting 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	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

Next steps: Generate code with df_agg_bookings | View recommended plots | New interactive sheet

Converting 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	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

Next steps: Generate code with df_agg_bookings | View recommended plots | New interactive sheet

```
df_bookings.head()
```

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

```
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: 760.8+ 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	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

Next steps: Generate code with df_agg_bookings | View recommended plots | New interactive sheet

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

room_category	occ_pct
RT1	57.889643
RT2	58.009756
RT3	58.028213
RT4	59.277925

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	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	16558	1-May-22	RT1	18	19.0	94.74	RT1	Standard

Next steps: Generate code with df | View recommended plots | New interactive sheet

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

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class
0	16559	1-May-22	RT1	25	30.0	83.33	Standard
1	19562	1-May-22	RT1	28	30.0	93.33	Standard
2	19563	1-May-22	RT1	23	30.0	76.67	Standard
3	16558	1-May-22	RT1	18	19.0	94.74	Standard

Next steps: Generate code with df | View recommended plots | New interactive sheet

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

room_class	occ_pct
Elite	58.009756
Premium	58.028213
Presidential	59.277925
Standard	57.889643

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

```
57.88964285714285
```

2. Printing 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

Next steps: Generate code with df_hotels | View recommended plots | New interactive sheet

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

	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

Next steps: Generate code with df | View recommended plots | New interactive sheet

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

city	occ_pct
Bangalore	56.332376
Delhi	61.507341
Hyderabad	58.120652
Mumbai	57.909181

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

Next steps: Generate code with df_date | View recommended plots | New interactive sheet

```
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	occ_pct	room_class	property_name	category	city
0	19563	10-May-22	RT3	15	29.0	51.72	Premium	Atliq Palace	Business	Bangalore

Next steps: Generate code with df | View recommended plots | New interactive sheet

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

day_type	occ_pct
weekeday	50.88
weekend	72.34

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)
```

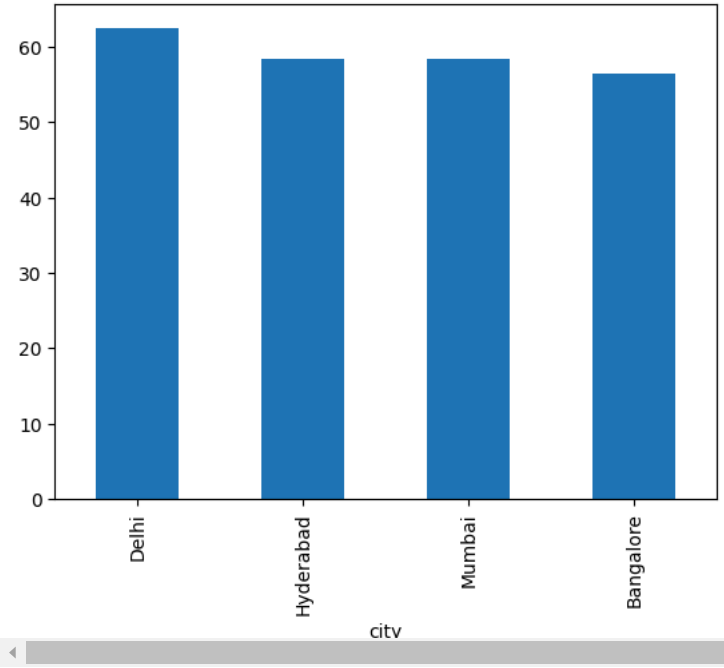
	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	ci
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury	Muml

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

city	occ_pct
Delhi	62.47
Hyderabad	58.46
Mumbai	58.38
Bangalore	56.44

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

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



5: We got new data for the month of august, Appending 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	check_in_date	mmm yy	week no	day_type	successful_bookin
0	16559	Atliq Exotica	Luxury	Mumbai	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	

Next steps: Generate code with df_august | View recommended plots | New interactive sheet

```
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	occ_pct	room_class	property_name	category	ci
6494	17558	31-Jul-22	RT4	3	6.0	50.0	Presidential	Atliq Grands	Luxury	Muml
6495	19563	31-Jul-22	RT4	3	6.0	50.0	Presidential	Atliq Palace	Business	Bangalc
6496	17561	31-Jul-22	RT4	3	4.0	75.0	Presidential	Atliq Blu	Luxury	Muml
6497	16559	01-Aug-22	RT1	30	30.0	NaN	Standard	Atliq Exotica	Luxury	Muml
6498	19562	01-Aug-22	RT1	21	30.0	NaN	Standard	Atliq Bay	Luxury	Bangalc

```
latest_df.shape
```

```
(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. Printing revenue realized per city

```
df_bookings.head()
```

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

```
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

Next steps: Generate code with df_hotels | View recommended plots | New interactive sheet

```
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	no_guests	room_category	booking_platform	ratings_g
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	

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

city	revenue_realized
Bangalore	420383550
Delhi	294404488
Hyderabad	325179310
Mumbai	668569251

7. Printing 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	weekday
2	03-May-22	May 22	W 19	weekday

Next steps: Generate code with df_date | View recommended plots | New interactive sheet

```
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	no_guests	room_category	booking_platform	ratings_g
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	

```
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)
```

```
<ipython-input-85-14d4af7ca1c1>:1: UserWarning: Could not infer format, so each element will be parsed individually, falling back to
  df_date["date"] = pd.to_datetime(df_date["date"])
```

	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

Next steps: Generate code with df_date | View recommended plots | New interactive sheet

```
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
```

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

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_g
0	May012216558RT12	16558	30-04-22	2022-01-05	2/5/2022	2.0	RT1	others	
1	May012216558RT15	16558	27-04-22	2022-01-05	2/5/2022	4.0	RT1	direct online	
2	May012216558RT16	16558	1/5/2022	2022-01-05	3/5/2022	2.0	RT1	others	
3	May012216558RT17	16558	28-04-22	2022-01-05	6/5/2022	2.0	RT1	others	

```
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	no_guests	room_category	booking_platform	ratings_g
0	May052216558RT11	16558	15-04-22	2022-05-05	7/5/2022	3.0	RT1	tripster	
1	May052216558RT12	16558	30-04-22	2022-05-05	7/5/2022	2.0	RT1	others	
2	May052216558RT13	16558	1/5/2022	2022-05-05	6/5/2022	3.0	RT1	direct offline	

Next steps: Generate code with df_bookings_all | View recommended plots | New interactive sheet

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

mmm yy	revenue_realized
Jul 22	389940912
Jun 22	377191229
May 22	408375641

Printing 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()
```

	revenue_realized
property_name	