==> 1. Data Import and Data Exploration

Read bookings data in a dataframe

In [2]:
```
! pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\devi sasi kala\appdata\local\progr
ams\python\python313\lib\site-packages (2.2.3)
Requirement already satisfied: numpy>=1.26.0 in c:\users\devi sasi kala\appdata\loca
l\programs\python\python313\lib\site-packages (from pandas) (2.2.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\devi sasi kala\app
data\local\programs\python\python313\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\devi sasi kala\appdata\local
\programs\python\python313\lib\site-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\devi sasi kala\appdata\loc
al\programs\python\python313\lib\site-packages (from pandas) (2025.1)
Requirement already satisfied: six>=1.5 in c:\users\devi sasi kala\appdata\local\pro
grams\python\python313\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.1
7.0)
[notice] A new release of pip is available: 24.3.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [6]:
```
import pandas as pd
df_bookings=pd.read_csv("fact_bookings.csv")
df_bookings.head()
```

Out[6]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0

Explore bookings data

In [7]:
```
df_bookings.shape
```

Out[7]:
```
(134590, 12)
```

In [13]:
```
df_bookings.room_category.unique()
```

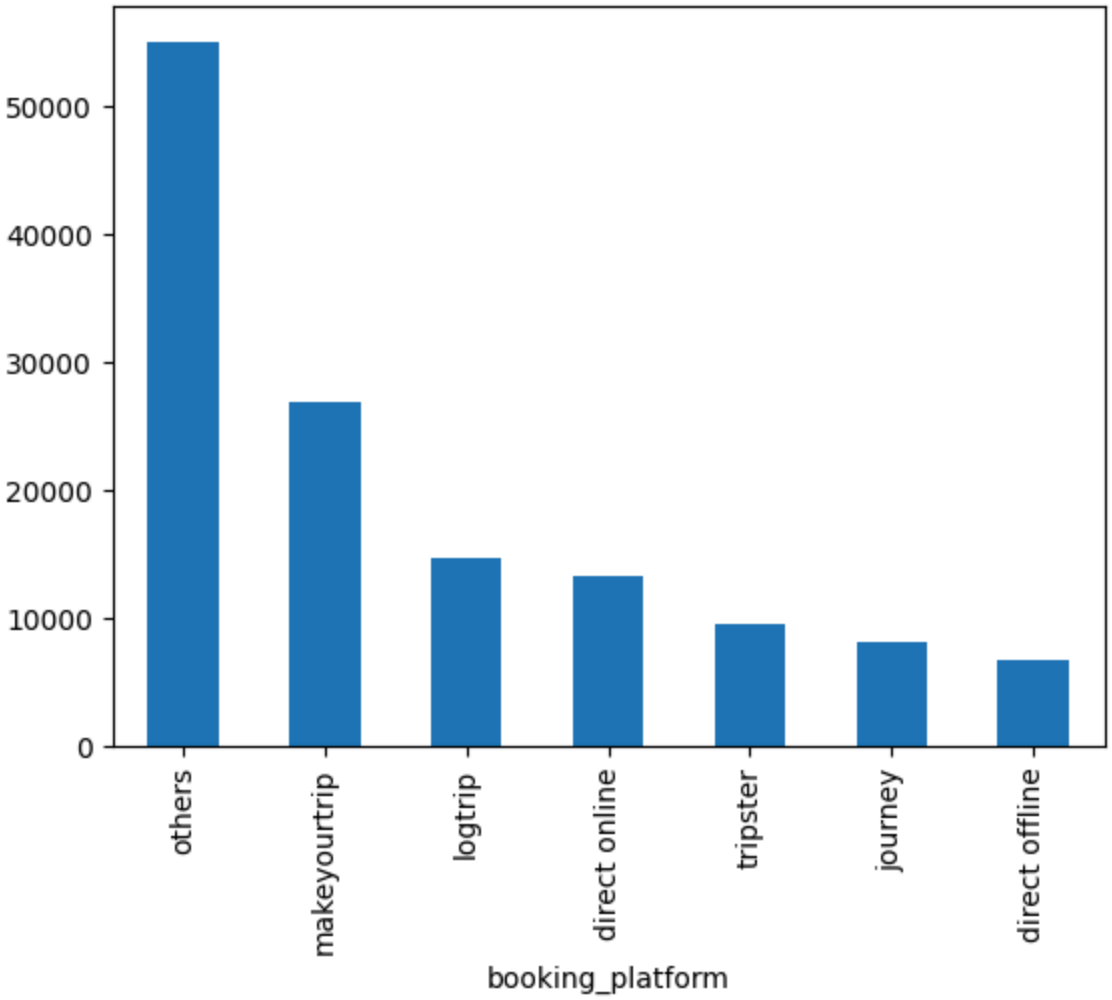
Out[13]:
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

In [14]:
```
df_bookings.booking_platform.unique()
```

Out[14]:
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

In [224…
```
df_bookings.booking_platform.value_counts().plot(kind="bar")
```

Out[224…
```
<Axes: xlabel='booking_platform'>
```



In [225…
```
df_bookings.describe()
```

Out[225…
```
np.float64(18061.113492830078)
```

In [12]:
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

Out[12]:
```
(np.int64(6500), np.int64(28560000))
```

Read rest of the files

In [24]:
```
df_date=pd.read_csv("dim_date.csv")
df_hotels=pd.read_csv("dim_hotels.csv")
df_rooms=pd.read_csv("dim_rooms.csv")
df_agg_bookings=pd.read_csv("fact_aggregated_bookings.csv")
df_fact_bookings=pd.read_csv("fact_bookings.csv")
```

In [25]:
```
df_hotels.shape
```

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

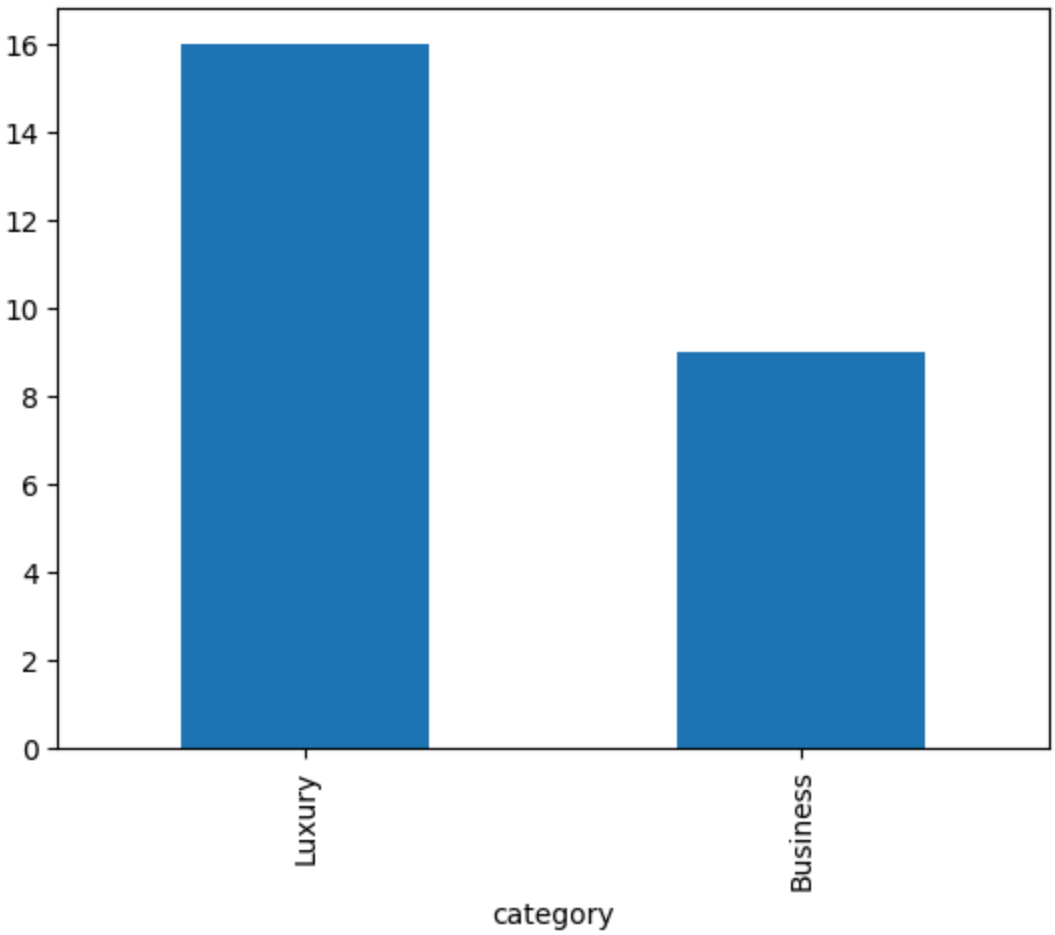
In [26]:
```
df_hotels.head(4)
```

Out[26]:

	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi
3	16561	Atliq Blu	Luxury	Delhi

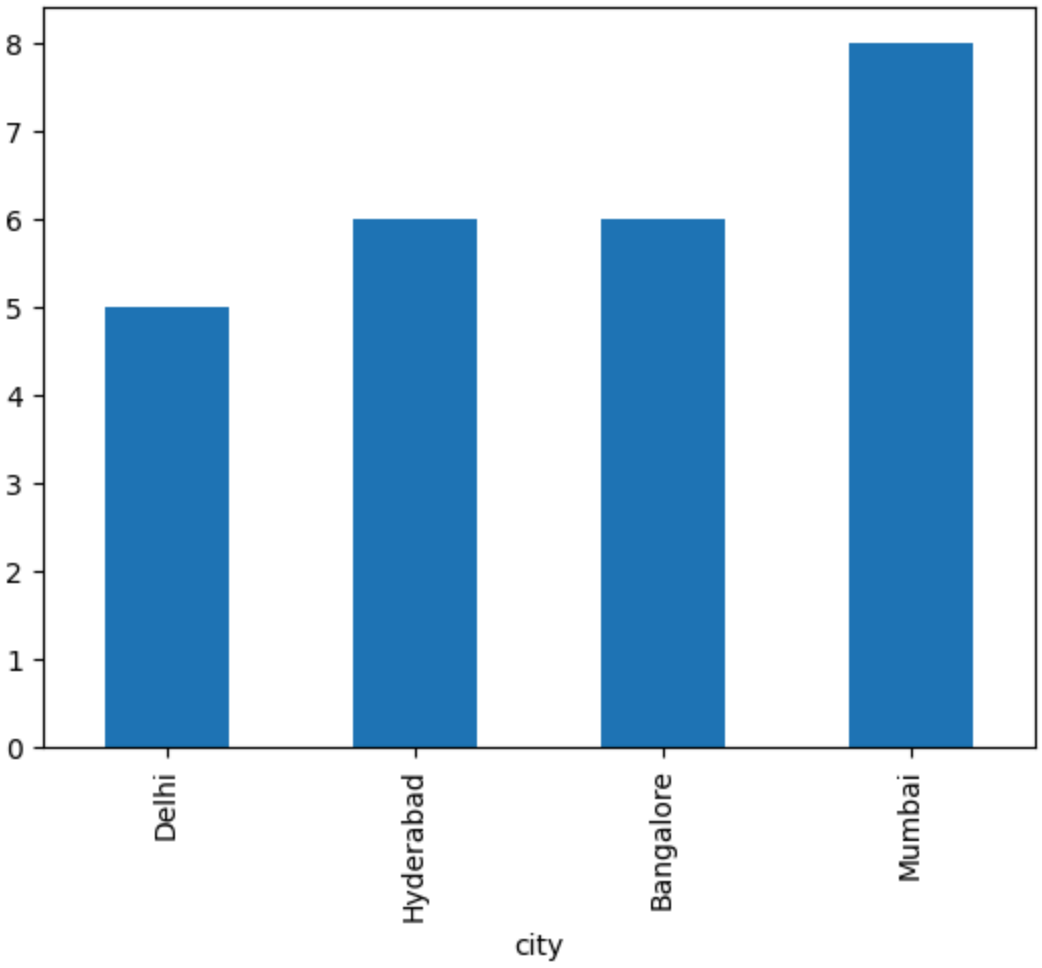
In [29]:
```
df_hotels.category.value_counts().plot(kind="bar")
```

Out[29]:
```
<Axes: xlabel='category'>
```



In [32]:
```
df_hotels.city.value_counts().sort_values().plot(kind="bar")
```

Out[32]:
```
<Axes: xlabel='city'>
```



Exercise: Explore aggregate bookings

In [220…
```
df_agg_bookings.shape
```

Out[220…
```
(9194, 6)
```

In [34]:
```
df_agg_bookings.head(4)
```

Out[34]:

	property_id	check_in_date	room_category	successful_bookings	capacity
0	16559	1-May-22	RT1	25	30.0
1	19562	1-May-22	RT1	28	30.0
2	19563	1-May-22	RT1	23	30.0
3	17558	1-May-22	RT1	30	19.0

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

In [35]:
```
df_agg_bookings.property_id.unique()
```

Out[35]:
```
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 [42]:
```
df_agg_bookings.groupby("property_id")["successful_bookings"].sum()
```

Out[42]:
```
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 [45]:
```
df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.capacity]
```

Out[45]:

	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 [64]:
```
df_agg_bookings.capacity.max()
```

Out[64]:
```
np.float64(50.0)
```

In [50]:
```
df_agg_bookings[df_agg_bookings.capacity==df_agg_bookings.capacity.max()]
```

Out[50]:

	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

In [15]:
```
df_bookings.describe()
```

Out[15]:

	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 [17]:
```
df_bookings[df_bookings.no_guests<=0]
```

Out[17]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	
17924	May122218559RT44	18559	12/5/2022	12/5/2022	14-05-22	
18020	May122218561RT22	18561	8/5/2022	12/5/2022	14-05-22	
18119	May122218562RT311	18562	5/5/2022	12/5/2022	17-05-22	
18121	May122218562RT313	18562	10/5/2022	12/5/2022	17-05-22	
56715	Jun082218562RT12	18562	5/6/2022	8/6/2022	13-06-22	
119765	Jul202219560RT220	19560	19-07-22	20-07-22	22-07-22	
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	

In [18]:
```
df_bookings[df_bookings.no_guests>=0]b
```

Out[18]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_g
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	
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	
...	...	...	...	...	...	
134584	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	

134578 rows × 12 columns

In [19]:
```
df_bookings.shape
```

Out[19]: (134590, 12)

(2) Outlier removal in revenue generated

In [20]:
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

Out[20]: (np.int64(6500), np.int64(28560000))

In [24]:
```
df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median()
```

Out[24]: (np.float64(15378.05412734973), np.float64(13500.0))

In [25]:
```
avg,std=df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median(
```

In [26]:
```
higher_limit = avg+3*std
higher_limit
```

Out[26]: np.float64(55878.05412734973)

In [27]:
```
lower_limit = avg-3*std
lower_limit
```

Out[27]: np.float64(-25121.94587265027)

In [28]:
```
df_bookings[df_bookings.revenue_generated<=0]
```

Out[28]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_c

In [33]:
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

Out[33]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	
111	May012216559RT32	16559	29-04-22	1/5/2022	2/5/2022	
315	May012216562RT22	16562	28-04-22	1/5/2022	4/5/2022	
562	May012217559RT118	17559	26-04-22	1/5/2022	2/5/2022	
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	

In [39]:
```
df_bookings[df_bookings.revenue_generated<=higher_limit]
df_bookings.shape
```

Out[39]: (134590, 12)

In [44]:
```
df_bookings.revenue_realized.describe()
```

Out[44]:
```
count    134590.000000
mean      12696.123256
std        6928.108124
min        2600.000000
25%        7600.000000
50%       11700.000000
75%       15300.000000
max       45220.000000
Name: revenue_realized, dtype: float64
```

In [45]:
```
higher_limit=df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.s
higher_limit
```

Out[45]:
```
np.float64(33480.44762788103)
```

In [48]:
```
df_bookings[df_bookings.revenue_realized>higher_limit]
```

Out[48]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	
139	May012216559RT43	16559	1/5/2022	1/5/2022	2/5/2022	
143	May012216559RT47	16559	28-04-22	1/5/2022	3/5/2022	
149	May012216559RT413	16559	24-04-22	1/5/2022	7/5/2022	
222	May012216560RT45	16560	30-04-22	1/5/2022	3/5/2022	
...	...	...	...	...	...	
134331	Jul312219560RT412	19560	31-07-22	31-07-22	1/8/2022	
134467	Jul312219562RT45	19562	28-07-22	31-07-22	1/8/2022	
134474	Jul312219562RT412	19562	25-07-22	31-07-22	6/8/2022	
134581	Jul312217564RT42	17564	31-07-22	31-07-22	1/8/2022	
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	

1300 rows × 12 columns

In [50]:
```
df_bookings[df_bookings.room_category=="RT4"].revenue_realized.describe()
```

Out[50]:
```
count    16073.000000
mean     23440.103652
std       9048.865206
min       7600.000000
25%      19000.000000
50%      26600.000000
75%      32300.000000
max      45220.000000
Name: revenue_realized, dtype: float64
```

In [60]:
```
23440 + 3*9048
```

Out[60]:
```
50584
```

In [59]:
```
df_bookings.isnull().sum()
```

Out[59]:
```
booking_id               0
property_id              0
booking_date             0
check_in_date            0
checkout_date            0
no_guests                3
room_category            0
booking_platform         0
ratings_given        77907
booking_status           0
revenue_generated        0
revenue_realized         0
dtype: int64
```

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 [3]:
```
import pandas as pd
df_agg_bookings=pd.read_csv("fact_aggregated_bookings.csv")
df_agg_bookings.head()
```

Out[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
3	17558	1-May-22	RT1	30	19.0
4	16558	1-May-22	RT1	18	19.0

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 [67]:
```
df_agg_bookings[df_agg_bookings.capacity.isna()]
```

Out[67]:

	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 [16]:
```
df_agg_bookings.capacity.median()
```

Out[16]: np.float64(25.0)

In [6]:
```
df_agg_bookings.capacity.fillna(df_agg_bookings.capacity.median(), inplace=True)
```

```
C:\Users\DEVI SASI KALA\AppData\Local\Temp\ipykernel_23260\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)
```

In [5]:
```
df_agg_bookings.loc[[8,15]]
```

Out[5]:

	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

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

Out[7]: (9200, 5)

In [81]:
```
df_agg_bookings[df_agg_bookings.successful_bookings<=df_agg_bookings.capacity]
df_agg_bookings.shape
```

Out[81]: (9194, 5)

==> 3. Data Transformation

Create occupancy percentage column

In [18]:
```
df_agg_bookings.head()
```

Out[18]:

	property_id	check_in_date	room_category	successful_bookings	capacity
0	16559	1-May-22	RT1	25	30.0
1	19562	1-May-22	RT1	28	30.0
2	19563	1-May-22	RT1	23	30.0
3	17558	1-May-22	RT1	30	19.0
4	16558	1-May-22	RT1	18	19.0

In [19]:
```
df_agg_bookings["occ_pct"]=df_agg_bookings["successful_bookings"]/df_agg_bookings["
df_agg_bookings.head()
```

Out[19]:

	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
3	17558	1-May-22	RT1	30	19.0	1.578947
4	16558	1-May-22	RT1	18	19.0	0.947368

In [20]:
```
df_agg_bookings["occ_pct"]=df_agg_bookings["occ_pct"].apply(lambda x: round(x*100,2
df_agg_bookings.head()
```

Out[20]:

	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

==> 4. Insights Generation

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

In [92]:
```
df_agg_bookings.head()
```

Out[92]:

	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
4	16558	1-May-22	RT1	18	19.0	94.74
5	17560	1-May-22	RT1	28	40.0	70.00

In [96]:
```
df_agg_bookings.groupby("room_category")["occ_pct"].mean()
```

Out[96]:
```
room_category
RT1    57.889643
RT2    58.009756
RT3    58.028213
RT4    59.277925
Name: occ_pct, dtype: float64
```

In [8]:
```
df_rooms=pd.read_csv("dim_rooms.csv")
df_rooms
```

Out[8]:

	room_id	room_class
0	RT1	Standard
1	RT2	Elite
2	RT3	Premium
3	RT4	Presidential

In [112…
```
df=pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_on="room_id")
df.head()
```

Out[112…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	roon
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	16558	1-May-22	RT1	18	19.0	94.74	
4	17560	1-May-22	RT1	28	40.0	70.00	

In [113…
```
df.groupby("room_class")["occ_pct"].mean()
```

Out[113…
```
room_class
Elite           58.009756
Premium         58.028213
Presidential    59.277925
Standard        57.889643
Name: occ_pct, dtype: float64
```

In [114…
```
df.drop("room_id", axis=1, inplace=True)
df.head()
```

Out[114…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	roon
0	16559	1-May-22	RT1	25	30.0	83.33	St
1	19562	1-May-22	RT1	28	30.0	93.33	St
2	19563	1-May-22	RT1	23	30.0	76.67	St
3	16558	1-May-22	RT1	18	19.0	94.74	St
4	17560	1-May-22	RT1	28	40.0	70.00	St

In [119…
```
df[df.room_class=="Standard"].occ_pct.mean()
```

Out[119…
```
np.float64(57.88964285714285)
```

2. Print average occupancy rate per city

In [9]:
```
df_hotels=pd.read_csv("dim_hotels.csv")
df_hotels.head()
```

Out[9]:

	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi
3	16561	Atliq Blu	Luxury	Delhi
4	16562	Atliq Bay	Luxury	Delhi

In [16]:
```
df=pd.merge(df_agg_bookings,df_hotels,left_on="property_id", right_on="property_id"
df.head()
```

Out[16]:

	property_id	check_in_date	room_category	successful_bookings	capacity	property_nam
0	16559	1-May-22	RT1	25	30.0	Atliq Exotic
1	19562	1-May-22	RT1	28	30.0	Atliq Ba
2	19563	1-May-22	RT1	23	30.0	Atliq Palac
3	17558	1-May-22	RT1	30	19.0	Atliq Grand
4	16558	1-May-22	RT1	18	19.0	Atliq Grand

In [129…
```
df.groupby("city")["occ_pct"].mean()
```

Out[129…
```
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?

In [12]:
```
df_date=pd.read_csv("dim_date.csv")
df_date.head()
```

Out[12]:

	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
3	04-May-22	May 22	W 19	weekeday
4	05-May-22	May 22	W 19	weekeday

In [13]:
```
df=pd.merge(df_agg_bookings,df_date,left_on="check_in_date", right_on="date")
df.head(4)
```

Out[13]:

	property_id	check_in_date	room_category	successful_bookings	capacity	date	mmm yy
0	19563	10-May-22	RT3	15	29.0	10-May-22	May 22
1	18560	10-May-22	RT1	19	30.0	10-May-22	May 22
2	19562	10-May-22	RT1	18	30.0	10-May-22	May 22
3	19563	10-May-22	RT1	16	30.0	10-May-22	May 22

In [134…
```
df.groupby("day_type")["occ_pct"].mean().round(2)
```

Out[134…
```
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

In [140…
```
df_june_22 = df[df["mmm yy"]=="Jun 22"]
df_june_22.head(4)
```

Out[140…

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	c
2200	16559	10-Jun-22	RT1	20	30.0	66.67	J
2201	19562	10-Jun-22	RT1	19	30.0	63.33	J
2202	19563	10-Jun-22	RT1	17	30.0	56.67	J
2203	17558	10-Jun-22	RT1	9	19.0	47.37	J

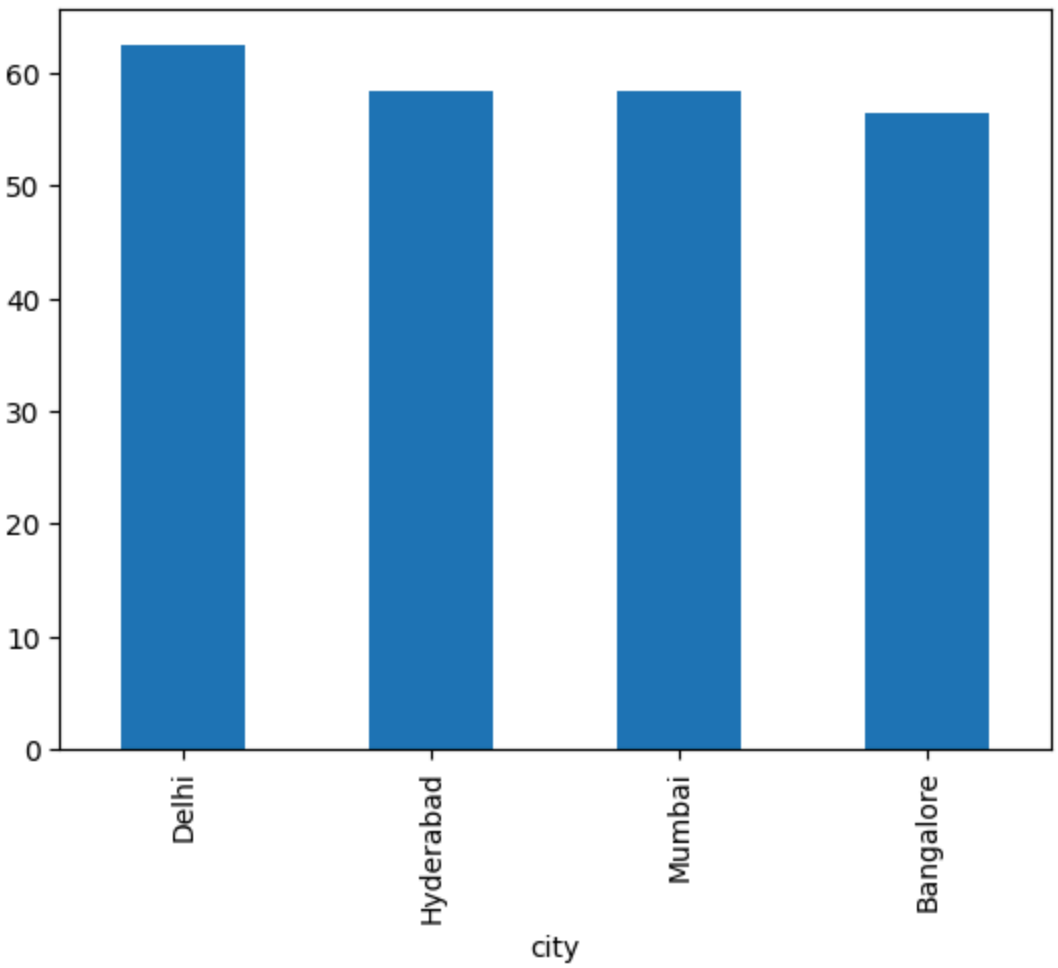
In [142…
```
df=pd.merge(df_hotels,df_june_22,left_on="property_id", right_on="property_id")
df.head()
```

Out[142...

	property_id	property_name	category	city	check_in_date	room_category	successful_l
0	16558	Atliq Grands	Luxury	Delhi	10-Jun-22	RT1	
1	16558	Atliq Grands	Luxury	Delhi	10-Jun-22	RT2	
2	16558	Atliq Grands	Luxury	Delhi	10-Jun-22	RT3	
3	16558	Atliq Grands	Luxury	Delhi	10-Jun-22	RT4	
4	16558	Atliq Grands	Luxury	Delhi	11-Jun-22	RT1	

In [146...
```
df.groupby("city")["occ_pct"].mean().round(2).sort_values(ascending=False).plot(kin
```

Out[146...
```
<Axes: xlabel='city'>
```



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

In [154…
```
df_august=pd.read_csv("new_data_august.csv")
df_august.head(4)
```

Out[154…

	property_id	property_name	category	city	room_category	room_class	check_in_
0	16559	Atliq Exotica	Luxury	Mumbai	RT1	Standard	01-Au
1	19562	Atliq Bay	Luxury	Bangalore	RT1	Standard	01-Au
2	19563	Atliq Palace	Business	Bangalore	RT1	Standard	01-Au
3	19558	Atliq Grands	Luxury	Bangalore	RT1	Standard	01-Au

In [155…
```
df_august.columns
```

Out[155…
```
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 [159…
```
df.columns
```

Out[159…
```
Index(['property_id', 'property_name', 'category', 'city', 'check_in_date',
       'room_category', 'successful_bookings', 'capacity', 'occ_pct', 'date',
       'mmm yy', 'week no', 'day_type'],
      dtype='object')
```

In [157…
```
df_august.shape
```

Out[157…
```
(7, 13)
```

In [158…
```
df.shape
```

Out[158…
```
(2099, 13)
```

In [161…
```
new_df=pd.concat([df,df_august], ignore_index=True, axis=0)
new_df.head(4)
```

Out[161…

	property_id	property_name	category	city	check_in_date	room_category	successful_l
0	16558	Atliq Grands	Luxury	Delhi	10-Jun-22	RT1	
1	16558	Atliq Grands	Luxury	Delhi	10-Jun-22	RT2	
2	16558	Atliq Grands	Luxury	Delhi	10-Jun-22	RT3	
3	16558	Atliq Grands	Luxury	Delhi	10-Jun-22	RT4	

In [162…
```
new_df.shape
```

Out[162… `(2106, 15)`

6. Print revenue realized per city

In [21]:
```
df_bookings=pd.read_csv("fact_aggregated_bookings.csv")
df_bookings.head(4)
```

Out[21]:

	property_id	check_in_date	room_category	successful_bookings	capacity
0	16559	1-May-22	RT1	25	30.0
1	19562	1-May-22	RT1	28	30.0
2	19563	1-May-22	RT1	23	30.0
3	17558	1-May-22	RT1	30	19.0

In [22]:
```
df_bookings.head(4)
```

Out[22]:

	property_id	check_in_date	room_category	successful_bookings	capacity
0	16559	1-May-22	RT1	25	30.0
1	19562	1-May-22	RT1	28	30.0
2	19563	1-May-22	RT1	23	30.0
3	17558	1-May-22	RT1	30	19.0

In [18]:
```
df_hotels.head(4)
```

Out[18]:

	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi
3	16561	Atliq Blu	Luxury	Delhi

In [23]:
```
df_bookings_all=pd.merge(df_bookings, df_hotels, left_on="property_id", right_on="p
df_bookings_all.head(4)
```

Out[23]:

	property_id	check_in_date	room_category	successful_bookings	capacity	property_nam
0	16559	1-May-22	RT1	25	30.0	Atliq Exotic
1	19562	1-May-22	RT1	28	30.0	Atliq Ba
2	19563	1-May-22	RT1	23	30.0	Atliq Palac
3	17558	1-May-22	RT1	30	19.0	Atliq Grand

In [170…
```
df.groupby("city")["revenue_realized"].sum()
```

Out[170…
```
city
Bangalore    420397050
Delhi        294500318
Hyderabad    325232870
Mumbai       668640991
Name: revenue_realized, dtype: int64
```

7. Print month by month revenue

In [24]:
```
df_date=pd.read_csv("dim_date.csv")
df_date.head(3)
```

Out[24]:

	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 [176…
```
df_date["mmm yy"].unique()
```

Out[176…
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

In [177…
```
df.head(4)
```

Out[177…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0

In [178…

```
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 [25]:

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

```
C:\Users\DEVI SASI KALA\AppData\Local\Temp\ipykernel_23260\173964601.py:1: UserWarni
ng: Could not infer format, so each element will be parsed individually, falling bac
k to `dateutil`. To ensure parsing is consistent and as-expected, please specify a f
ormat.
  df_date["date"] = pd.to_datetime(df_date["date"])
```

Out[25]:

	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

In [194…

```
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     datetime64[ns]
 1   mmm yy    92 non-null     object        
 2   week no   92 non-null     object        
 3   day_type  92 non-null     object        
dtypes: datetime64[ns](1), object(3)
memory usage: 3.0+ KB
```

In [197…

```
df_bookings_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134590 entries, 0 to 134589
Data columns (total 15 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   booking_id         134590 non-null  object 
 1   property_id        134590 non-null  int64  
 2   booking_date       134590 non-null  object 
 3   check_in_date      134590 non-null  object 
 4   checkout_date      134590 non-null  object 
 5   no_guests          134587 non-null  float64
 6   room_category      134590 non-null  object 
 7   booking_platform   134590 non-null  object 
 8   ratings_given      56683 non-null   float64
 9   booking_status     134590 non-null  object 
 10  revenue_generated  134590 non-null  int64  
 11  revenue_realized   134590 non-null  int64  
 12  property_name      134590 non-null  object 
 13  category           134590 non-null  object 
 14  city               134590 non-null  object 
dtypes: float64(2), int64(3), object(10)
memory usage: 15.4+ MB
```

In [202…

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

Out[202…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0

In [203…

```
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on="check_in_date", right
df_bookings_all.head(4)
```

Out[203…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests
0	May012216558RT11	16558	27-04-22	2022-05-01	2/5/2022	-3.0
1	May012216558RT12	16558	30-04-22	2022-05-01	2/5/2022	2.0
2	May012216558RT13	16558	28-04-22	2022-05-01	4/5/2022	2.0
3	May012216558RT14	16558	28-04-22	2022-05-01	2/5/2022	-2.0

In [206…

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

Out[206…
```
mmm yy
Jul 22    243180932
Jun 22    229644140
May 22    234516453
Name: revenue_realized, dtype: int64
```

Exercise-1. Print revenue realized per hotel type

In [207…
```
df_bookings.head(3)
```

Out[207…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0

In [208…
```
df_hotels.head(3)
```

Out[208…

	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 [210…
```
df=pd.merge(df_bookings, df_hotels,on="property_id")
df.head(3)
```

Out[210…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0

In [213…
```
df.groupby("property_name")["revenue_realized"].sum().sort_values()
```

Out[213…
```
property_name
Atliq Seasons     66125495
Atliq Grands     211532764
Atliq Bay        260051178
Atliq Blu        260855522
Atliq City       285811939
Atliq Palace     304081863
Atliq Exotica    320312468
Name: revenue_realized, dtype: int64
```

Exercise-2 Print average rating per city

In [215…
```
df.groupby("city")["ratings_given"].mean().round(2).sort_values()
```

Out[215…
```
city
Bangalore    3.41
Mumbai       3.65
Hyderabad    3.66
Delhi        3.78
Name: ratings_given, dtype: float64
```

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

In [216…
```
df_bookings.head(3)
```

Out[216…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0

In [219…
```
df_bookings.groupby("booking_platform")["revenue_realized"].sum().plot(kind="pie")
```

Out[219…
```
<Axes: ylabel='revenue_realized'>
```