AtliQ Hotels

Exploratory Data Analysis in the Hospitality Domain Project

BY: AKHIL KUMAR • ASPIRING DATA ANALYST

Project Overview

- End-to-end Python-based hotel data analysis
- Involved cleaning, merging, transforming datasets
- KPI calculations: Occupancy %, ADR, Revenue
- Insights generated for business decisions

Datasets Used

- • dim_date.csv
- • dim_hotels.csv
- • dim_rooms.csv
- • fact_aggregated_bookings.csv
- • fact_bookings.csv

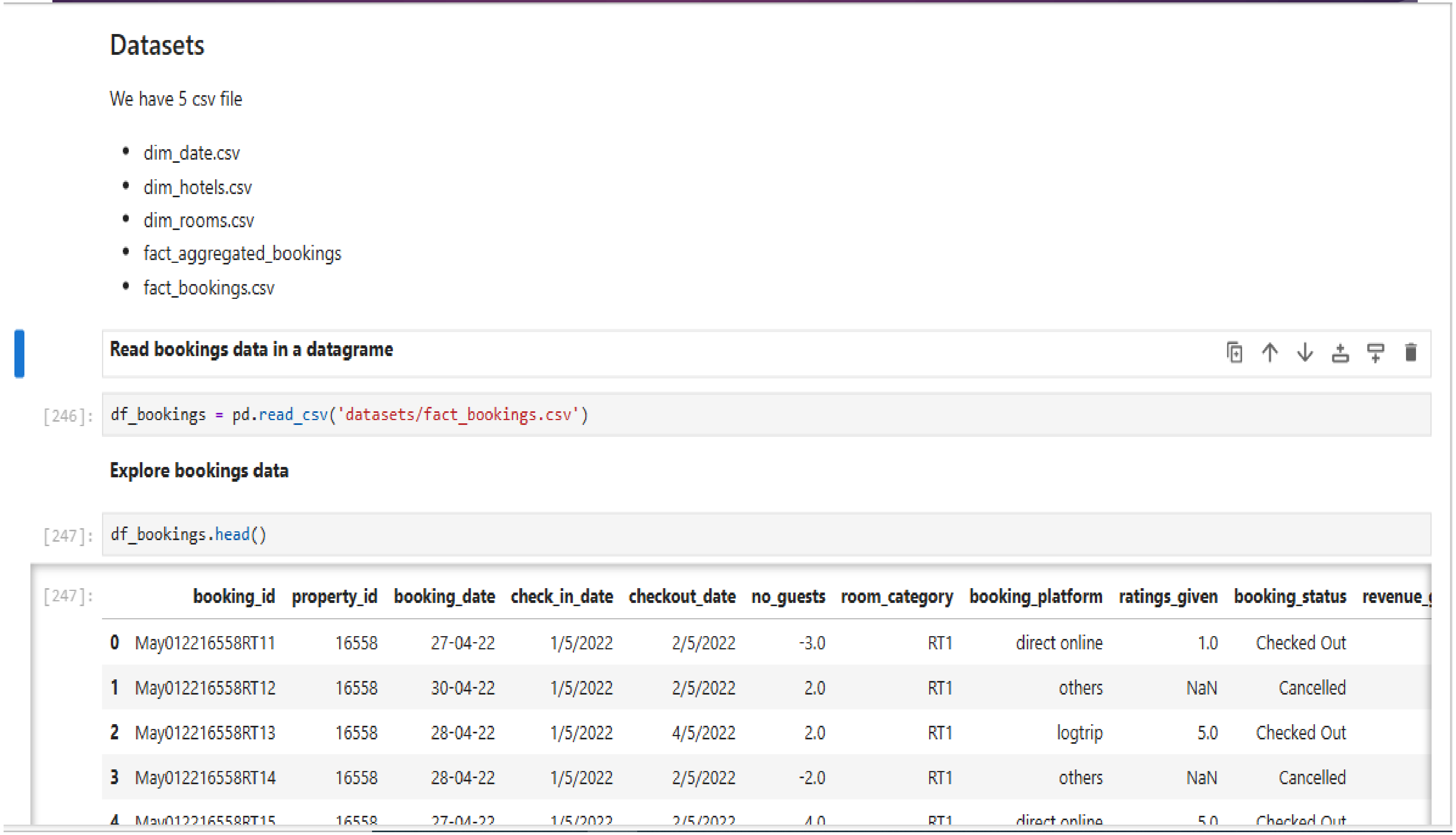
Analysis Workflow

- 1. Importing Data
- 2. Exploratory Data Analysis (EDA)
- 3. Data Cleaning & Merging
- 4. KPI Creation
- 5. Trend Analysis
- 6. Final Insights

Tech Stack Used

- Python (pandas, numpy, matplotlib)
- Jupyter Notebook
- Excel/CSV Inputs

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

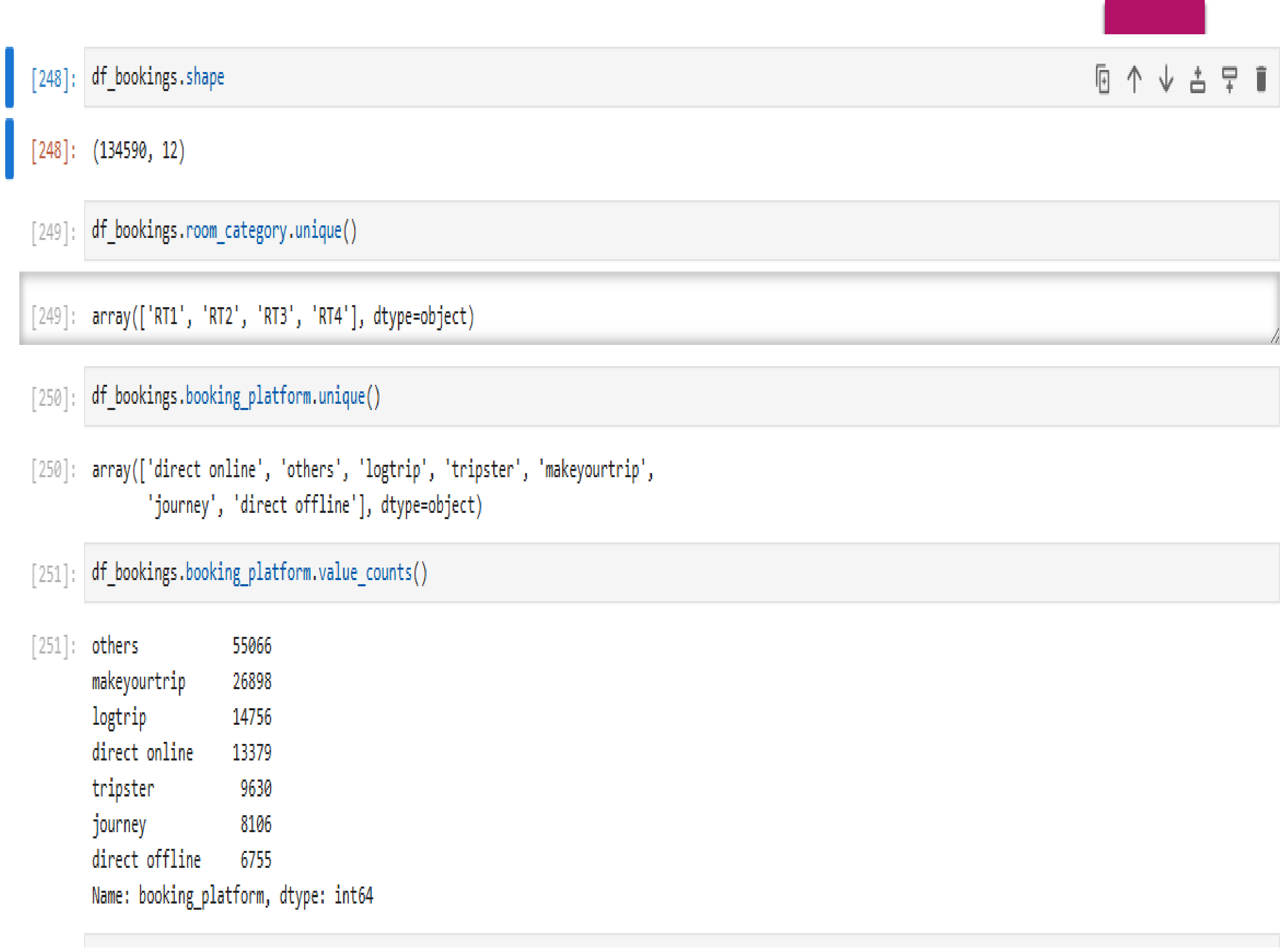
```
[246]: df_bookings = pd.read_csv('datasets/fact_bookings.csv')
```

Explore bookings data

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

[247]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_
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	4.0	RT1	direct online	5.0	Checked Out	



```
[248]: df_bookings.shape
```

```
[248]: (134590, 12)
```

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

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

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

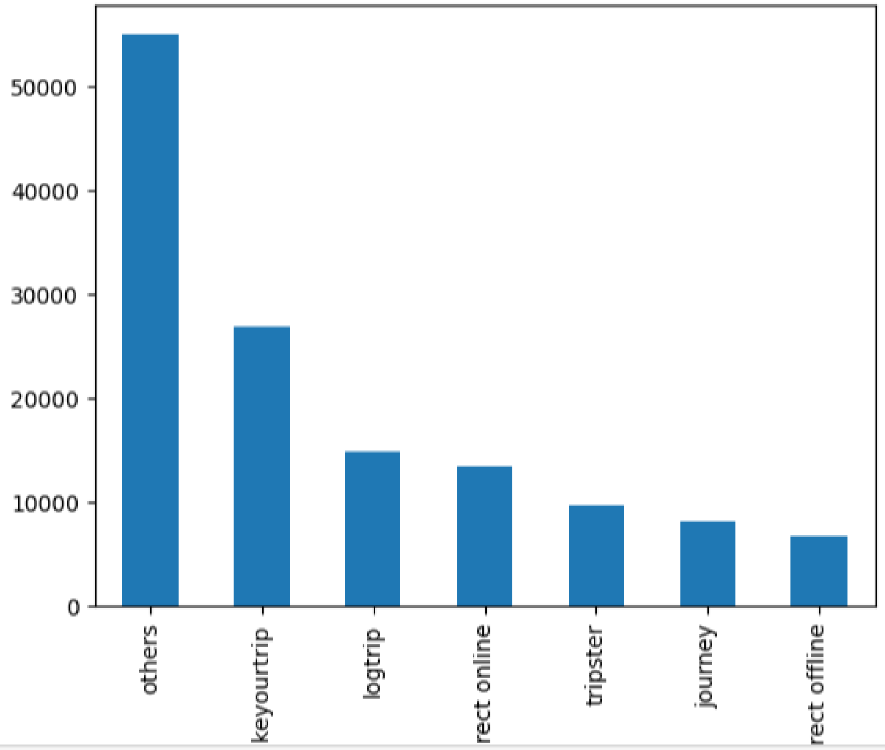
```
[250]: array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
               'journey', 'direct offline'], dtype=object)
```

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

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

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

```
[252]: <AxesSubplot: >
```



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

[253]:

	property_id	no_guests	ratings_given	revenue_generated	revenue_realized
count	134590.000000	134587.000000	56683.000000	1.345900e+05	134590.000000
mean	18061.113493	2.036170	3.619004	1.537805e+04	12696.123256
std	1093.055847	1.034885	1.235009	9.303604e+04	6928.108124
min	16558.000000	-17.000000	1.000000	6.500000e+03	2600.000000
25%	17558.000000	1.000000	3.000000	9.900000e+03	7600.000000
50%	17564.000000	2.000000	4.000000	1.350000e+04	11700.000000
75%	18563.000000	2.000000	5.000000	1.800000e+04	15300.000000
max	19563.000000	6.000000	5.000000	2.856000e+07	45220.000000

Read rest of the files

```
[254]: 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')
```

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

```
[255]: (25, 4)
```

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

[256]:

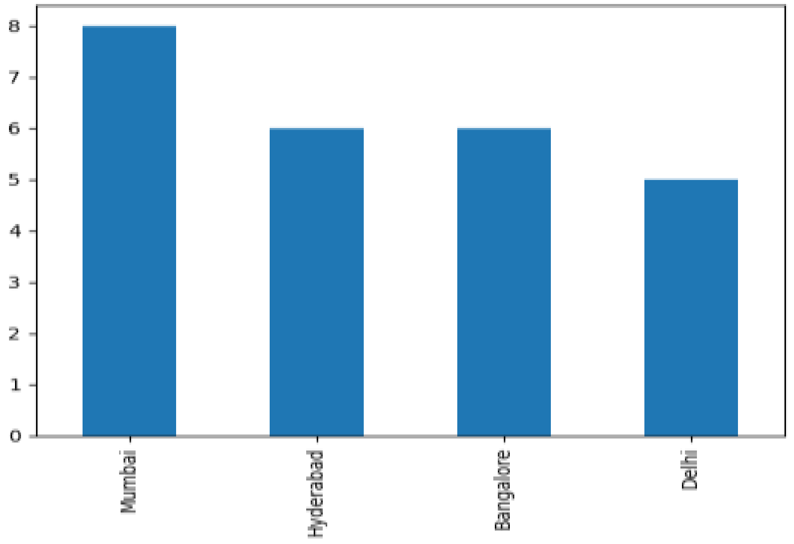
	property_id	property_name	category	city
0	16558	Atliq Grands	Luxury	Delhi
1	16559	Atliq Exotica	Luxury	Mumbai
2	16560	Atliq City	Business	Delhi

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

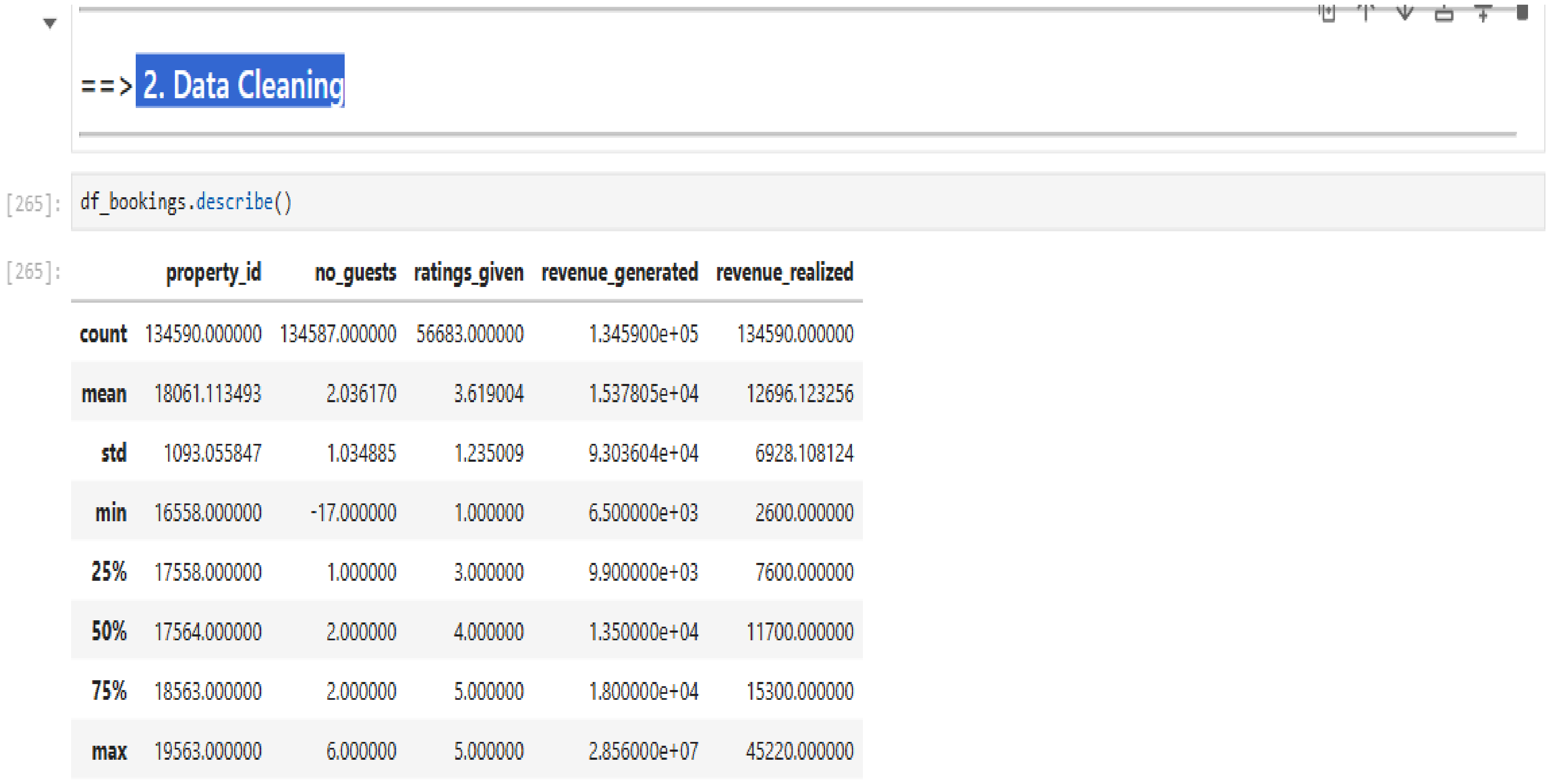
```
[257]: Luxury      16
        Business     9
        Name: category, dtype: int64
```

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

```
[258]: <AxesSubplot: >
```



2. Data Cleaning

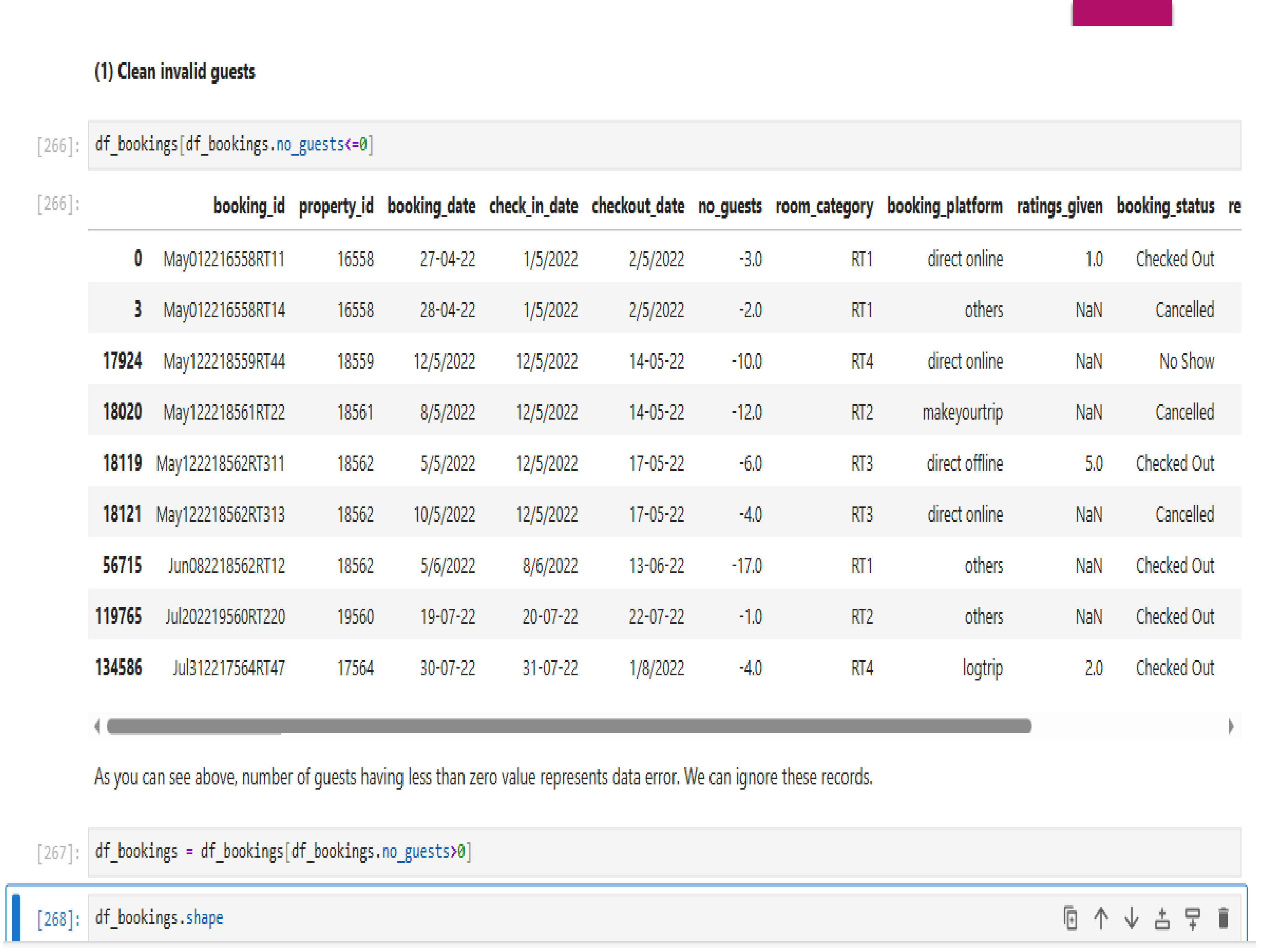


==> 2. Data Cleaning

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

[265]:

	property_id	no_guests	ratings_given	revenue_generated	revenue_realized
count	134590.000000	134587.000000	56683.000000	1.345900e+05	134590.000000
mean	18061.113493	2.036170	3.619004	1.537805e+04	12696.123256
std	1093.055847	1.034885	1.235009	9.303604e+04	6928.108124
min	16558.000000	-17.000000	1.000000	6.500000e+03	2600.000000
25%	17558.000000	1.000000	3.000000	9.900000e+03	7600.000000
50%	17564.000000	2.000000	4.000000	1.350000e+04	11700.000000
75%	18563.000000	2.000000	5.000000	1.800000e+04	15300.000000
max	19563.000000	6.000000	5.000000	2.856000e+07	45220.000000



(1) Clean invalid guests

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

[266]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	re
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	1.0	Checked Out	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	NaN	Cancelled	
17924	May122218559RT44	18559	12/5/2022	12/5/2022	14-05-22	-10.0	RT4	direct online	NaN	No Show	
18020	May122218561RT22	18561	8/5/2022	12/5/2022	14-05-22	-12.0	RT2	makeyourtrip	NaN	Cancelled	
18119	May122218562RT311	18562	5/5/2022	12/5/2022	17-05-22	-6.0	RT3	direct offline	5.0	Checked Out	
18121	May122218562RT313	18562	10/5/2022	12/5/2022	17-05-22	-4.0	RT3	direct online	NaN	Cancelled	
56715	Jun082218562RT12	18562	5/6/2022	8/6/2022	13-06-22	-17.0	RT1	others	NaN	Checked Out	
119765	Jul202219560RT220	19560	19-07-22	20-07-22	22-07-22	-1.0	RT2	others	NaN	Checked Out	
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	-4.0	RT4	logtrip	2.0	Checked Out	

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

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

```
[268]: df_bookings.shape
```

(2) Outlier removal in revenue generated

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

```
[269]: (6500, 28560000)
```

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

```
[270]: (15378.036937686695, 13500.0)
```

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

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

```
[272]: 294498.50173207896
```

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

```
[273]: -263742.4278567056
```

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

[274]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_generate

[275]:
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

[275]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	re
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	5.0	Checked Out	
111	May012216559RT32	16559	29-04-22	1/5/2022	2/5/2022	6.0	RT3	direct online	NaN	Checked Out	
315	May012216562RT22	16562	28-04-22	1/5/2022	4/5/2022	2.0	RT2	direct offline	3.0	Checked Out	
562	May012217559RT118	17559	26-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancelled	
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	2.0	RT2	direct online	3.0	Checked Out	

[276]:
```
df_bookings = df_bookings[df_bookings.revenue_generated<=higher_limit]
df_bookings.shape
```

[276]:
```
(134573, 12)
```

[277]:
```
df_bookings.revenue_realized.describe()
```

[277]:
```
count    134573.000000
mean      12695.983585
std        6927.791692
min        2600.000000
25%        7600.000000
50%       11700.000000
75%       15300.000000
max       45220.000000
```

[278]:
```
higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std()
higher_limit
```

[278]:
```
33479.358661845814
```

[279]:
```
df_bookings[df_bookings.revenue_realized>higher_limit]
```

[279]:

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

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

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

```
[280]: count    16071.000000
       mean     23439.308444
       std       9048.599076
       min       7600.000000
       25%      19000.000000
       50%      26600.000000
       75%      32300.000000
       max      45220.000000
       Name: revenue_realized, dtype: float64
```

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

```
[281]: 50583
```

Here higher limit comes to be 50583 and in our dataframe above we can see that max value for revenue realized is 45220. Hence we can conclude that there is no outlier and we don't need to do any data cleaning on this particular column

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

[282]:

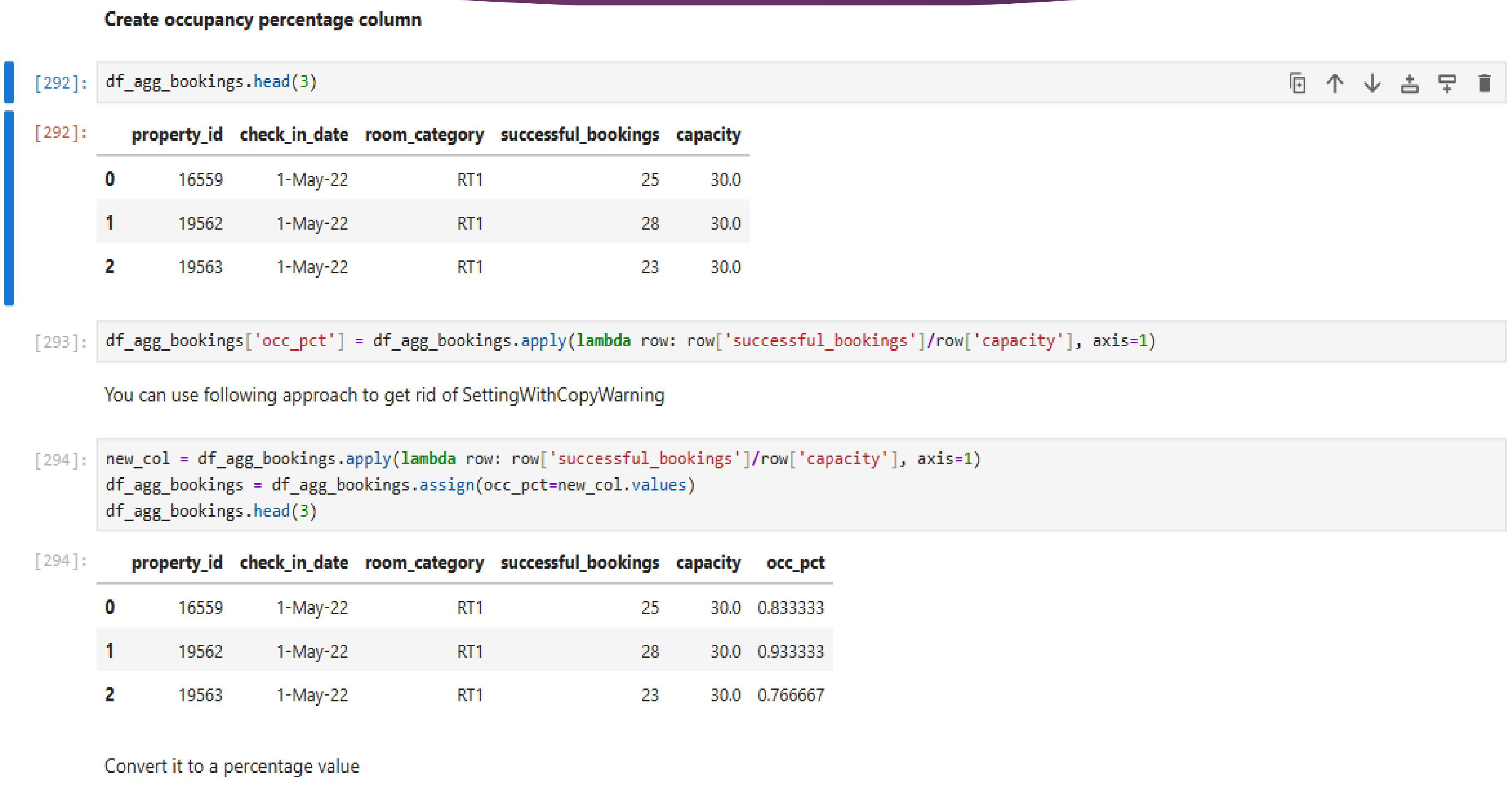
booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_generatec

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

```
[283]: 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

3. Data Transformation



Create occupancy percentage column

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

[292]:

	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

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

You can use following approach to get rid of SettingWithCopyWarning

```
[294]: 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)
```

[294]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
0	16559	1-May-22	RT1	25	30.0	0.833333
1	19562	1-May-22	RT1	28	30.0	0.933333
2	19563	1-May-22	RT1	23	30.0	0.766667

Convert it to a percentage value

[295]:
```
df_agg_bookings['occ_pct'] = df_agg_bookings['occ_pct'].apply(lambda x: round(x*100, 2))
df_agg_bookings.head(3)
```

[295]:

	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

[299]:
```
df_bookings.head()
```

[299]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancelled	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	Checked Out	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	Checked Out	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	Cancelled	
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	NaN	No Show	

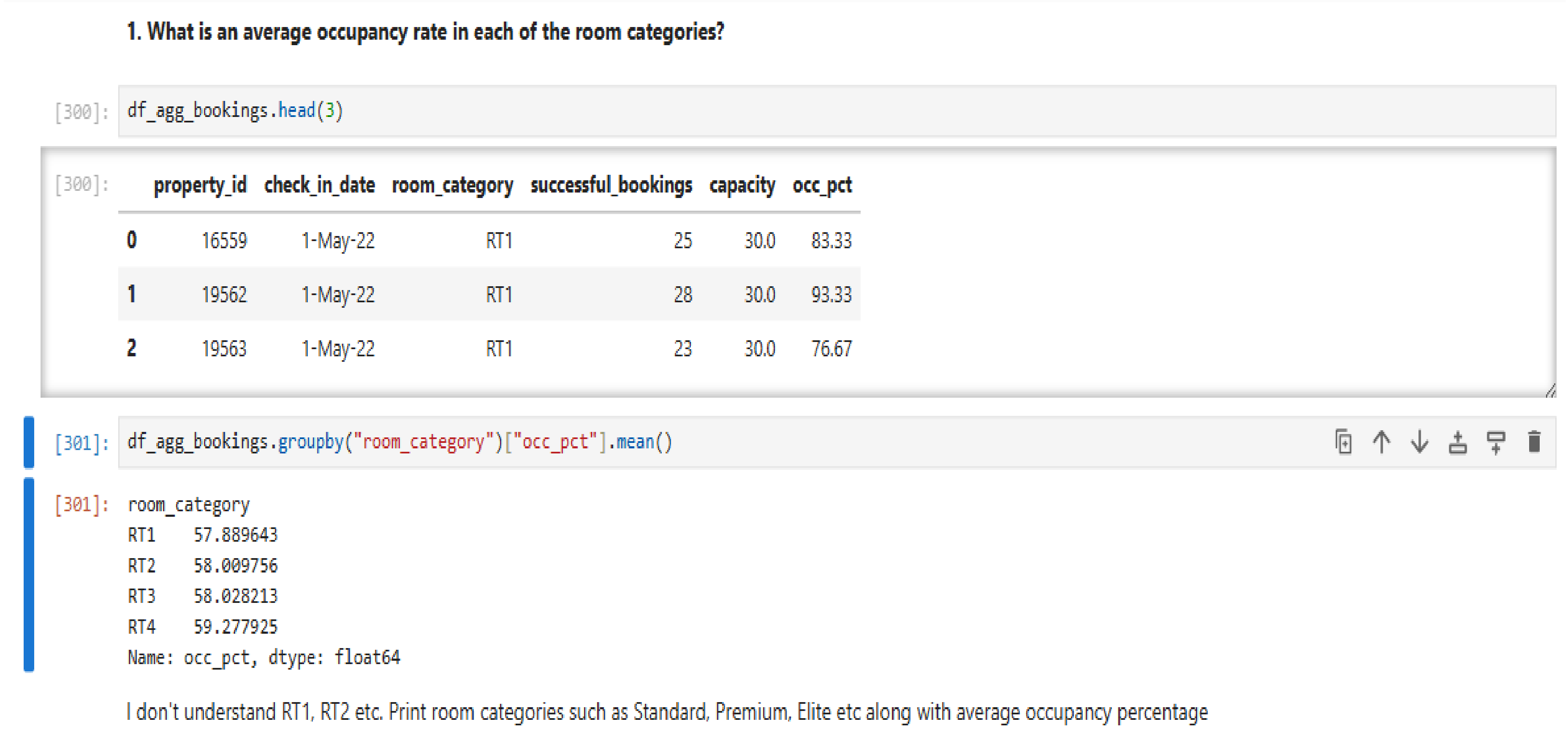
[297]:
```
df_agg_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9194 entries, 0 to 9199
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   property_id          9194 non-null   int64
 1   check_in_date        9194 non-null   object
 2   room_category        9194 non-null   object
 3   successful_bookings  9194 non-null   int64
 4   capacity             9194 non-null   float64
 5   occ_pct              9194 non-null   float64
dtypes: float64(2), int64(2), object(2)
memory usage: 502.8+ KB
```

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

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

4. Insights Generation



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

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

[300]:

	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

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

```
[301]: 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

[304]:
```
df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_on="room_id")
df.head(4)
```

[304]:

	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

[306]:
```
df.drop("room_id",axis=1, inplace=True)
df.head(4)
```

[306]:

	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

[306]:

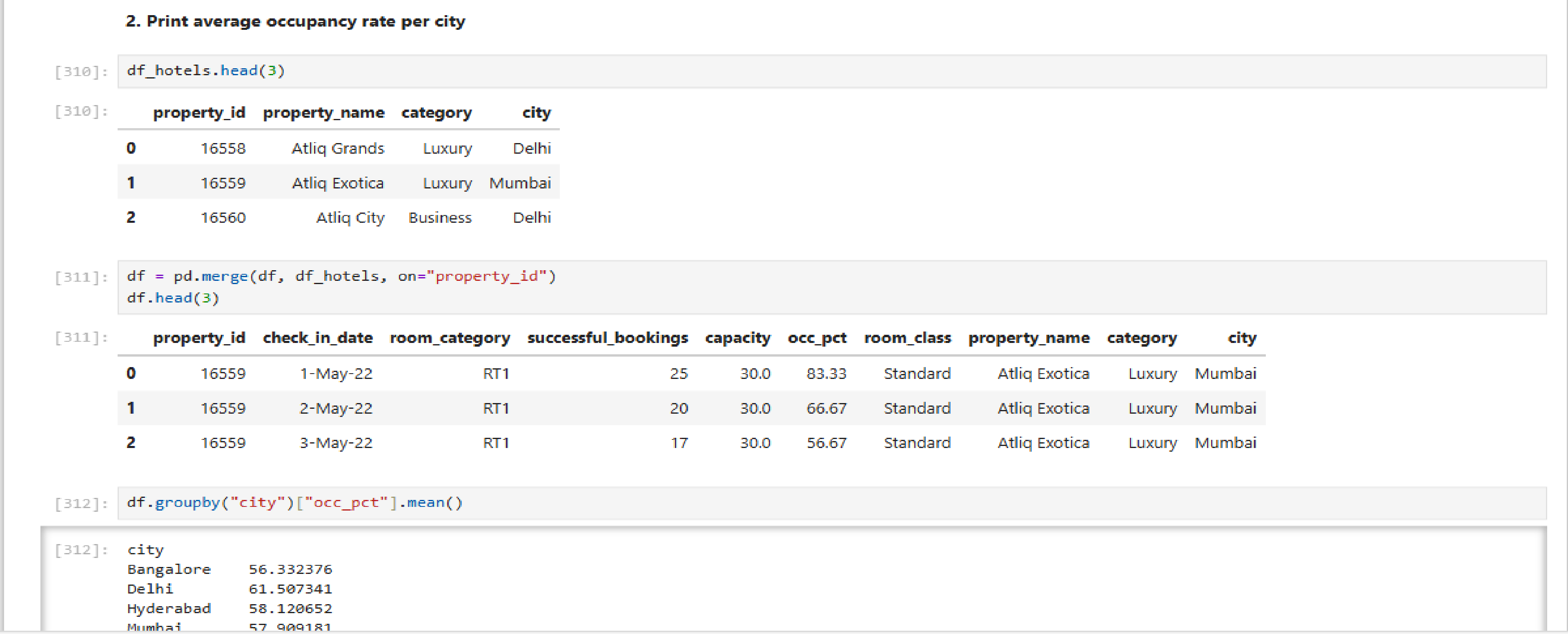
	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class
0	16559	1-May-22	RT1	25	30.0	83.33	Standard
1	19562	1-May-22	RT1	28	30.0	93.33	Standard
2	19563	1-May-22	RT1	23	30.0	76.67	Standard
3	16558	1-May-22	RT1	18	19.0	94.74	Standard

[308]:
```
df.groupby("room_class")["occ_pct"].mean()
```

[308]:
```
room_class
Elite           58.009756
Premium         58.028213
Presidential    59.277925
Standard        57.889643
Name: occ_pct, dtype: float64
```

[309]:
```
df[df.room_class=="Standard"].occ_pct.mean()
```

[309]:
```
57.88964285714285
```



2. Print average occupancy rate per city

[310]:
```
df_hotels.head(3)
```

[310]:

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

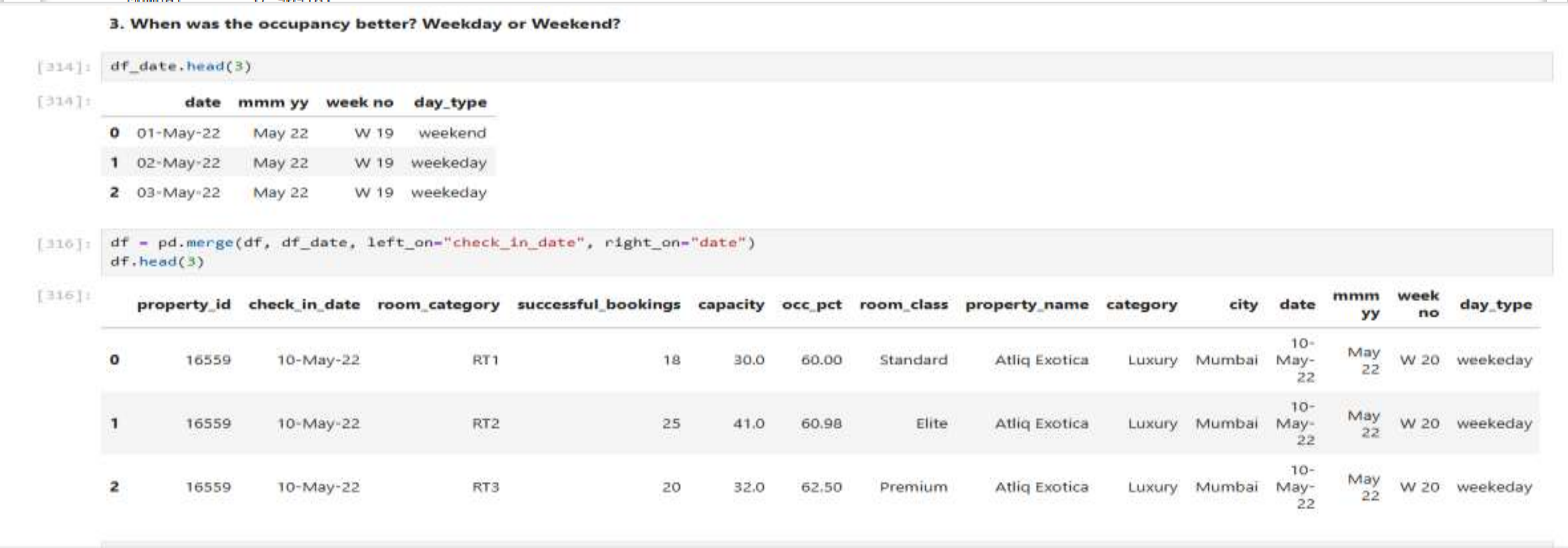
[311]:
```
df = pd.merge(df, df_hotels, on="property_id")
df.head(3)
```

[311]:

	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	16559	2-May-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury	Mumbai
2	16559	3-May-22	RT1	17	30.0	56.67	Standard	Atliq Exotica	Luxury	Mumbai

[312]:
```
df.groupby("city")["occ_pct"].mean()
```

[312]:
```
city
Bangalore    56.332376
Delhi        61.507341
Hyderabad    58.120652
Mumbai       57.909181
```



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

[314]:
```
df_date.head(3)
```

[314]:

	date	mmm yy	week no	day_type
0	01-May-22	May 22	W 19	weekend
1	02-May-22	May 22	W 19	weekeday
2	03-May-22	May 22	W 19	weekeday

[316]:
```
df = pd.merge(df, df_date, left_on="check_in_date", right_on="date")
df.head(3)
```

[316]:

	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
0	16559	10-May-22	RT1	18	30.0	60.00	Standard	Atliq Exotica	Luxury	Mumbai	10-May-22	May 22	W 20	weekeday
1	16559	10-May-22	RT2	25	41.0	60.98	Elite	Atliq Exotica	Luxury	Mumbai	10-May-22	May 22	W 20	weekeday
2	16559	10-May-22	RT3	20	32.0	62.50	Premium	Atliq Exotica	Luxury	Mumbai	10-May-22	May 22	W 20	weekeday

4: In the month of June, what is the occupancy for different cities

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

[323]:

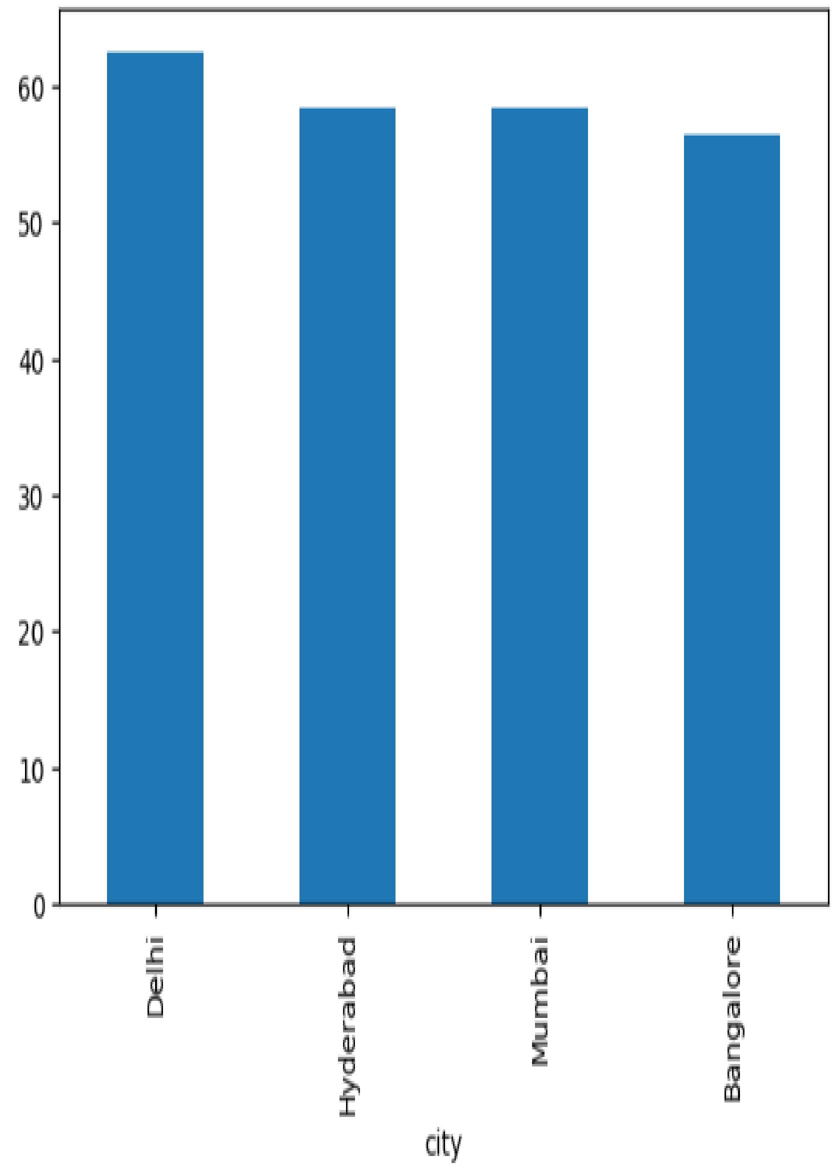
	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	date	mmm yy	week no	day_type
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury	Mumbai	10-Jun-22	Jun 22	W 24	weekeday
2201	16559	10-Jun-22	RT2	26	41.0	63.41	Elite	Atliq Exotica	Luxury	Mumbai	10-Jun-22	Jun 22	W 24	weekeday
2202	16559	10-Jun-22	RT3	20	32.0	62.50	Premium	Atliq Exotica	Luxury	Mumbai	10-Jun-22	Jun 22	W 24	weekeday
2203	16559	10-Jun-22	RT4	11	18.0	61.11	Presidential	Atliq Exotica	Luxury	Mumbai	10-Jun-22	Jun 22	W 24	weekeday

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

```
[324]: city
       Delhi        62.47
       Hyderabad    58.46
       Mumbai       58.38
       Bangalore    56.44
```

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

```
[327]: <AxesSubplot: xlabel='city'>
```



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

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

[329]:

	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	76.67

```
[334]: df_august.columns
```

```
[334]: Index(['property_id', 'property_name', 'category', 'city', 'room_category',
              'room_class', 'check_in_date', 'mmm yy', 'week no', 'day_type',
              'successful_bookings', 'capacity', 'occ%'],
             dtype='object')
```

```
[332]: df.columns
```

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

```
[337]: df_august.shape
```

```
[337]: (7, 13)
```

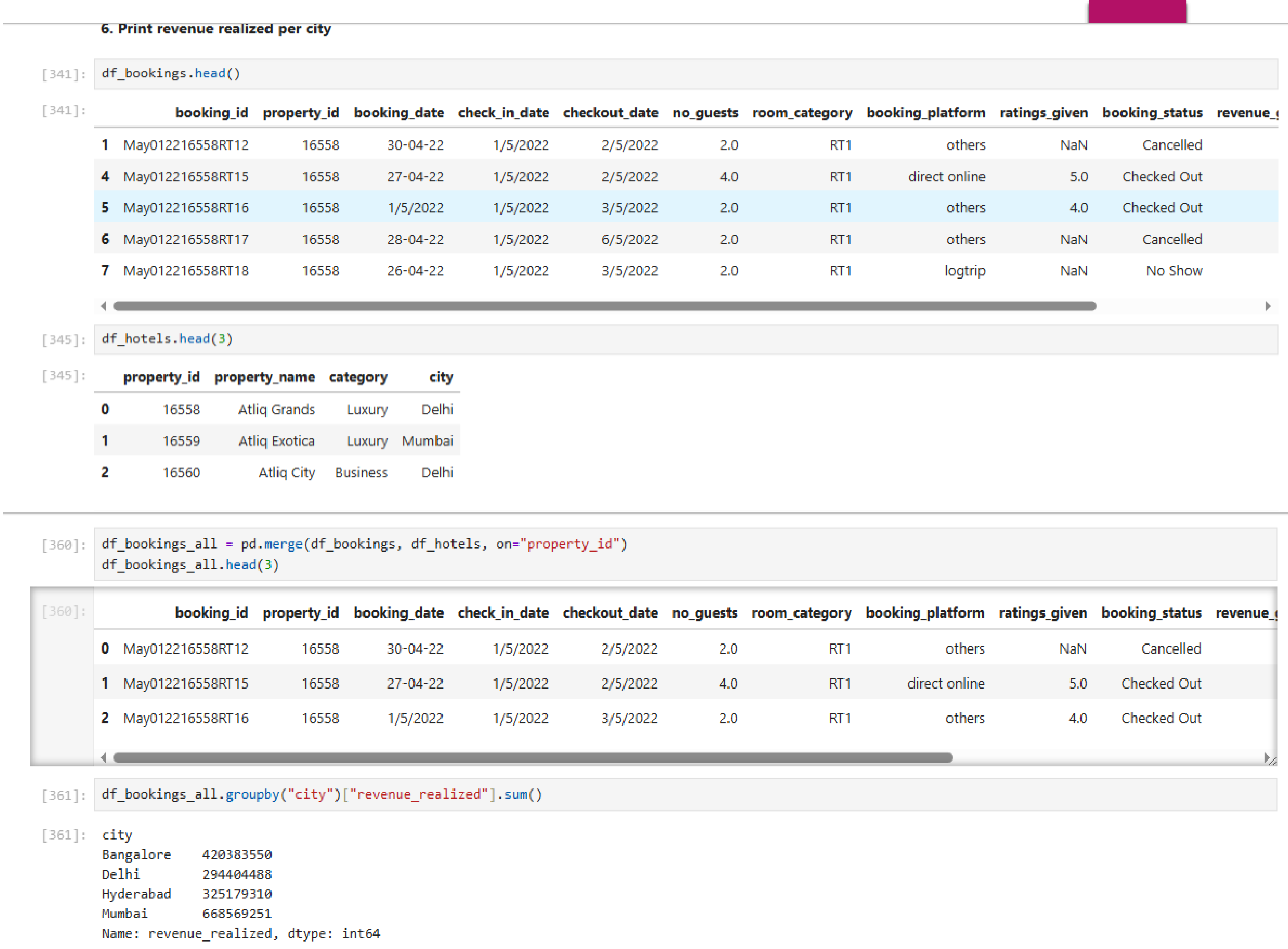
```
[338]: df.shape
```

```
[338]: (6497, 14)
```

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

[336]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	date	mmm yy	week no	day_ty
6494	16563	31-Jul-22	RT2	32	38.0	84.21	Elite	Atliq Palace	Business	Delhi	31-Jul-22	Jul 22	W 32	weeke
6495	16563	31-Jul-22	RT3	14	20.0	70.00	Premium	Atliq Palace	Business	Delhi	31-Jul-22	Jul 22	W 32	weeke
6496	16563	31-Jul-22	RT4	13	18.0	72.22	Presidential	Atliq Palace	Business	Delhi	31-Jul-22	Jul 22	W 32	weeke
6497	16559	01-Aug-22	RT1	30	30.0	NaN	Standard	Atliq Exotica	Luxury	Mumbai	NaN	Aug-22	W 32	weeked



6. Print revenue realized per city

[341]:
```python
df_bookings.head()
```

[341]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancelled	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	Checked Out	
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	Checked Out	
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	Cancelled	
7	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	NaN	No Show	

[345]:
```python
df_hotels.head(3)
```

[345]:

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

[360]:
```python
df_bookings_all = pd.merge(df_bookings, df_hotels, on="property_id")
df_bookings_all.head(3)
```

[360]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancelled	
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	Checked Out	
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	Checked Out	

[361]:
```python
df_bookings_all.groupby("city")["revenue_realized"].sum()
```

[361]:
```
city
Bangalore    420383550
Delhi        294404488
Hyderabad    325179310
Mumbai       668569251
Name: revenue_realized, dtype: int64
```

7. Print month by month revenue

[356]:
```
df_date.head(3)
```

[356]:

	date	mmm yy	week no	day_type
0	01-May-22	May 22	W 19	weekend
1	02-May-22	May 22	W 19	weekeday
2	03-May-22	May 22	W 19	weekeday

[357]:
```
df_date["mmm yy"].unique()
```

[357]:
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

[363]:
```
df_bookings_all.head(3)
```

[363]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancelled	
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0	Checked Out	
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2.0	RT1	others	4.0	Checked Out	

[364]:
```
df_date.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92 entries, 0 to 91
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   date      92 non-null     object
 1   mmm yy    92 non-null     object
 2   week no   92 non-null     object
 3   day_type  92 non-null     object
dtypes: object(4)
memory usage: 3.0+ KB
```

[365]:
```
df_date["date"] = pd.to_datetime(df_date["date"])
df_date.head(3)
```

[365]:

	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

[366]:
```
df_bookings_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 134573 entries, 0 to 134572
Data columns (total 15 columns):
 #   Column              Non-Null Count   Dtype
```

[367]:
```
df_bookings_all["check_in_date"] = pd.to_datetime(df_bookings_all["check_in_date"])
df_bookings_all.head(4)
```

[367]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_
0	May012216558RT12	16558	30-04-22	2022-01-05	2/5/2022	2.0	RT1	others	NaN	Cancelled	
1	May012216558RT15	16558	27-04-22	2022-01-05	2/5/2022	4.0	RT1	direct online	5.0	Checked Out	
2	May012216558RT16	16558	1/5/2022	2022-01-05	3/5/2022	2.0	RT1	others	4.0	Checked Out	
3	May012216558RT17	16558	28-04-22	2022-01-05	6/5/2022	2.0	RT1	others	NaN	Cancelled	

[368]:
```
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on="check_in_date", right_on="date")
df_bookings_all.head(3)
```

[368]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_
0	May052216558RT11	16558	15-04-22	2022-05-05	7/5/2022	3.0	RT1	tripster	5.0	Checked Out	
1	May052216558RT12	16558	30-04-22	2022-05-05	7/5/2022	2.0	RT1	others	NaN	Cancelled	
2	May052216558RT13	16558	1/5/2022	2022-05-05	6/5/2022	3.0	RT1	direct offline	5.0	Checked Out	

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

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

Exercise-1. Print revenue realized per hotel type

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

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

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

```
[352]: property_name
       Atliq Seasons     66086735
       Atliq Grands     211462134
       Atliq Bay        259996918
       Atliq Blu        260851922
       Atliq City       285798439
       Atliq Palace     304081863
       Atliq Exotica    320258588
       Name: revenue_realized, dtype: int64
```

Exercise-2 Print average rating per city

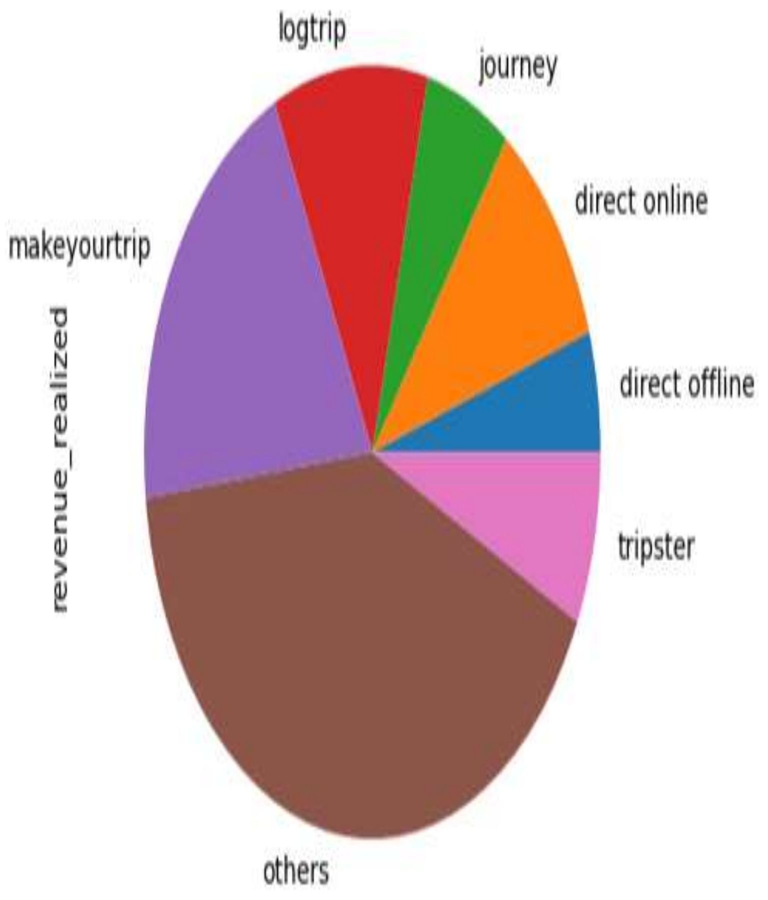
```
[354]: df_bookings_all.groupby("city")["ratings_given"].mean().round(2)
```

```
[354]: 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

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

```
[377]: <AxesSubplot: ylabel='revenue_realized'>
```



Key Insights & Findings

- • Identified occupancy trends across hotels
- • Analyzed booking patterns & seasonal variations
- • Revenue drivers highlighted using KPIs
- • Found data-quality issues and cleaned them

Conclusion

- The project demonstrates end-to-end analytical workflow
- Shows strong data cleaning + transformation skills
- Provides insights relevant for hotel business strategy

Connect with Me

- LinkedIn : https://www.linkedin.com/in/akhilkumar-ak/
- GitHub: https://github.com/akhilkumar-dataAnalyst
- Email: akhilkumar55077@gmail.com