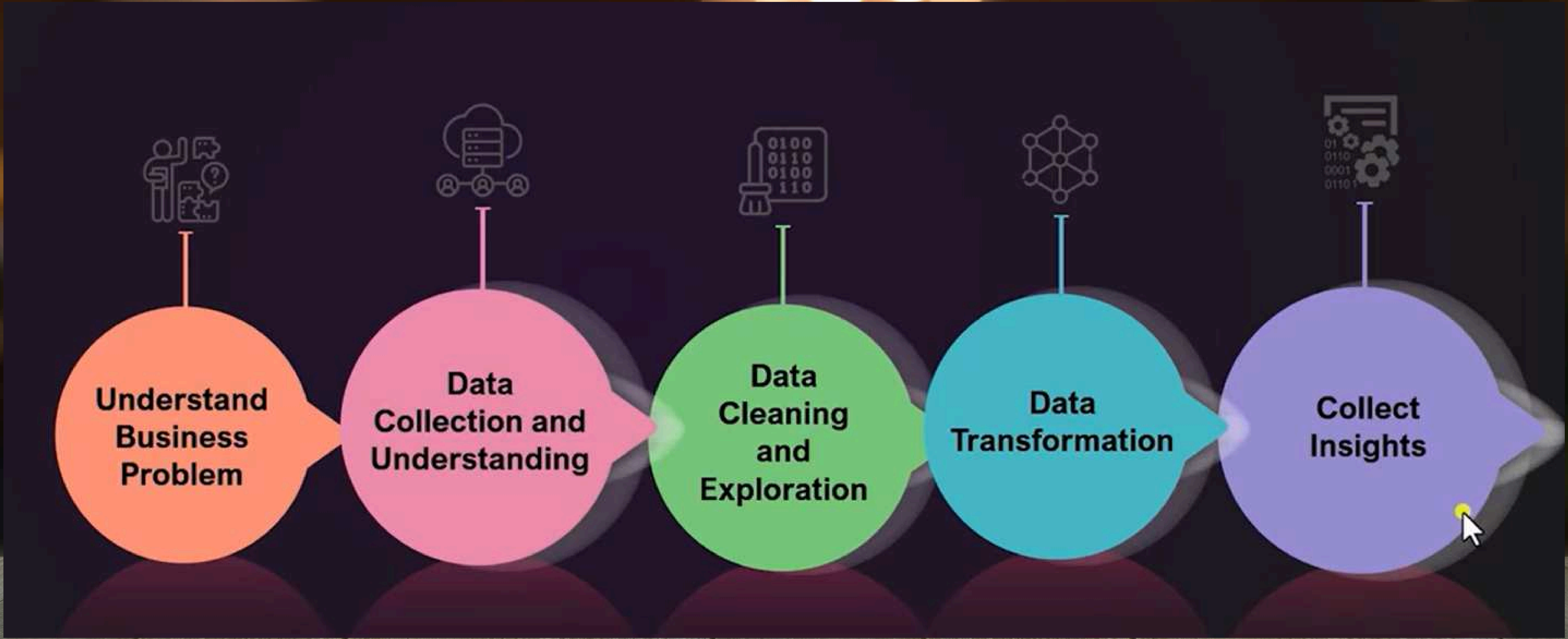
AtliQ Grands

PYTHON HOSPITALITY DOMAIN PROJECT

Insights Presented By
-Shimran Gupta

CONTENT

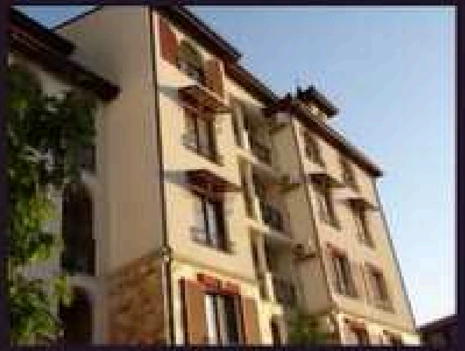
Project Steps

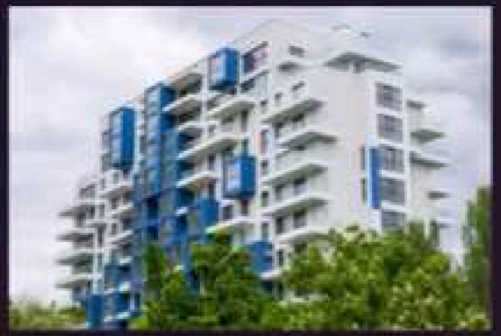

Understand Business Problem
Data Collection and Understanding
Data Cleaning and Exploration
Data Transformation
Collect Insights

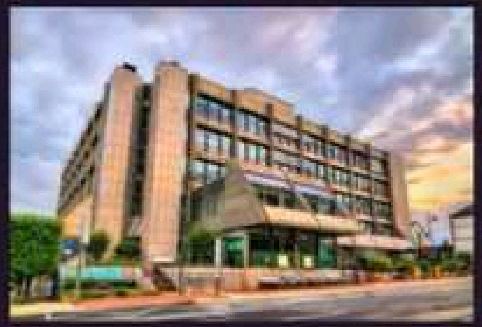
About Us

AtliQ Grand is a well-established hotel chain operating in major Indian cities such as Delhi, Mumbai, Bangalore, and Hyderabad. With over 20 years in the industry, the chain offers a diverse range of hotels and room categories to cater to various customer preferences.

Different Types of Hotels

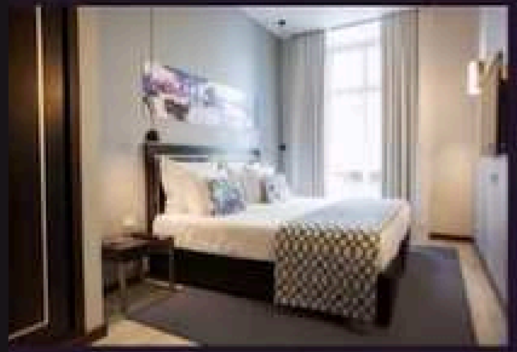

AtliQ Seasons

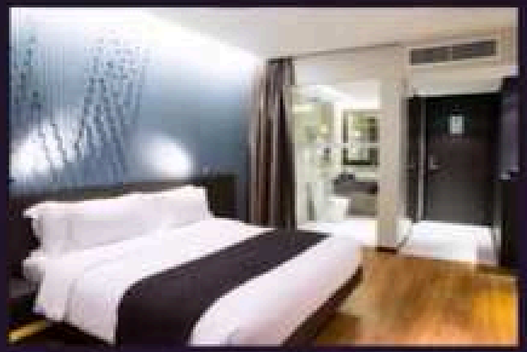

AtliQ Exotica

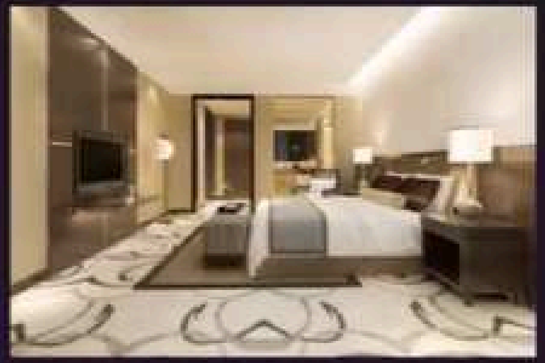

AtliQ Bay

AtliQ Palace

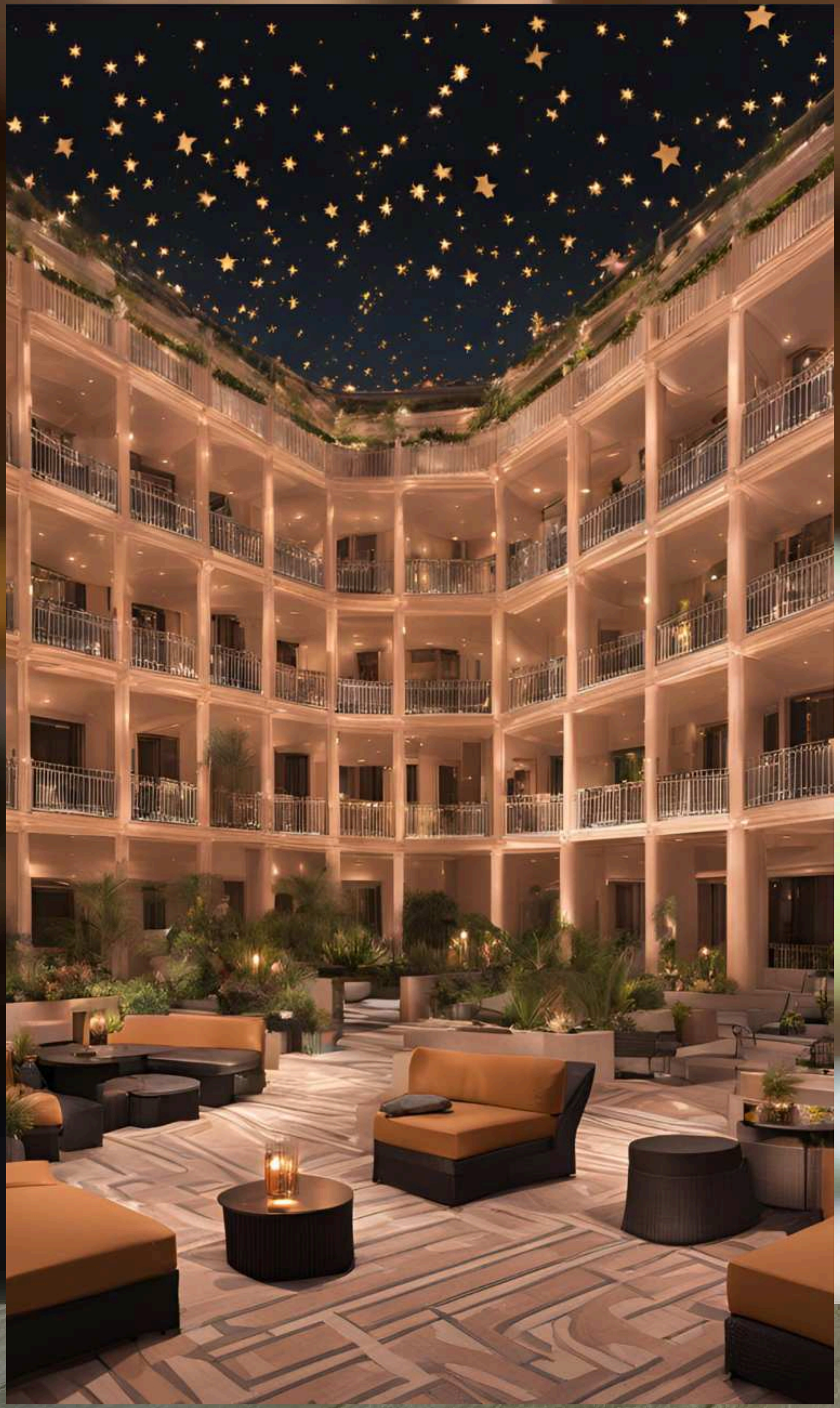
Different Types of Rooms


Standard


Elite


Premium

Presidential



Problem Statement

- **Challenges Faced:** AtliQ Grand, a prominent hotel chain with properties across India, is encountering difficulties in maintaining its competitive edge.
- **Revenue and Market Share Decline:** The company is seeing a decline in both revenue and market share, despite having multiple booking channels, including their own website and third-party platforms.
- **Booking Channels:** AtliQ Grand utilizes various booking channels, but these have not prevented the decline in financial performance.
- **Strategic Improvement:** A data-driven strategy is required to improve decision-making and tackle the issues affecting revenue and market share.

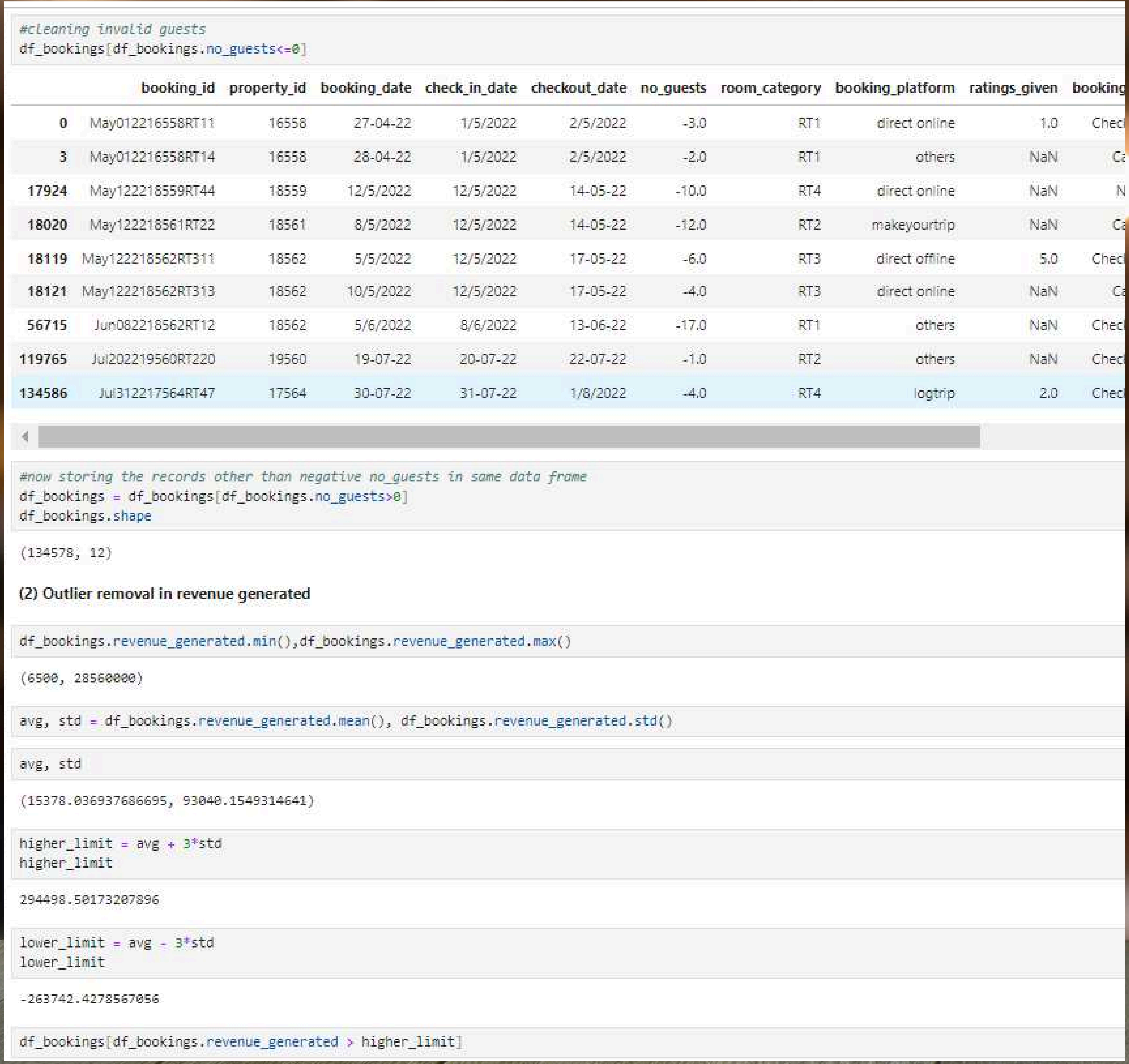
t of an
others,
ns can
shows,

Objectives

- Analyze Bookings Data: The project aims to analyze booking data from multiple sources to uncover insights.
- Enhance Revenue and Market Understanding: The goal is to enhance revenue streams, understand market dynamics, and regain a competitive position.
- Identify Key Factors: Leverage data analytics to identify factors contributing to revenue loss and assess current strategy effectiveness.
- Develop Actionable Recommendations: Provide recommendations to optimize booking processes and marketing efforts based on data insights.

DATA CLEANING

- Rectified invalid guest IDs where the number of guests was found to be negative.
- Conducted thorough checks for any null values in the dataset.
- Identified and removed outliers using the standard deviation method.



```
#cleaning 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	ratings_given	booking
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	1.0	Chec
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	NaN	Ca
17924	May122218559RT44	18559	12/5/2022	12/5/2022	14-05-22	-10.0	RT4	direct online	NaN	N
18020	May122218561RT22	18561	8/5/2022	12/5/2022	14-05-22	-12.0	RT2	makeyourtrip	NaN	Ca
18119	May122218562RT311	18562	5/5/2022	12/5/2022	17-05-22	-6.0	RT3	direct offline	5.0	Chec
18121	May122218562RT313	18562	10/5/2022	12/5/2022	17-05-22	-4.0	RT3	direct online	NaN	Ca
56715	Jun082218562RT12	18562	5/6/2022	8/6/2022	13-06-22	-17.0	RT1	others	NaN	Chec
119765	Jul202219560RT220	19560	19-07-22	20-07-22	22-07-22	-1.0	RT2	others	NaN	Chec
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	-4.0	RT4	logtrip	2.0	Chec

```
#now storing the records other than negative no_guests in same data frame
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)
```

```
avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.std()
```

```
avg, std
```

```
(15378.036937686695, 93040.1549314641)
```

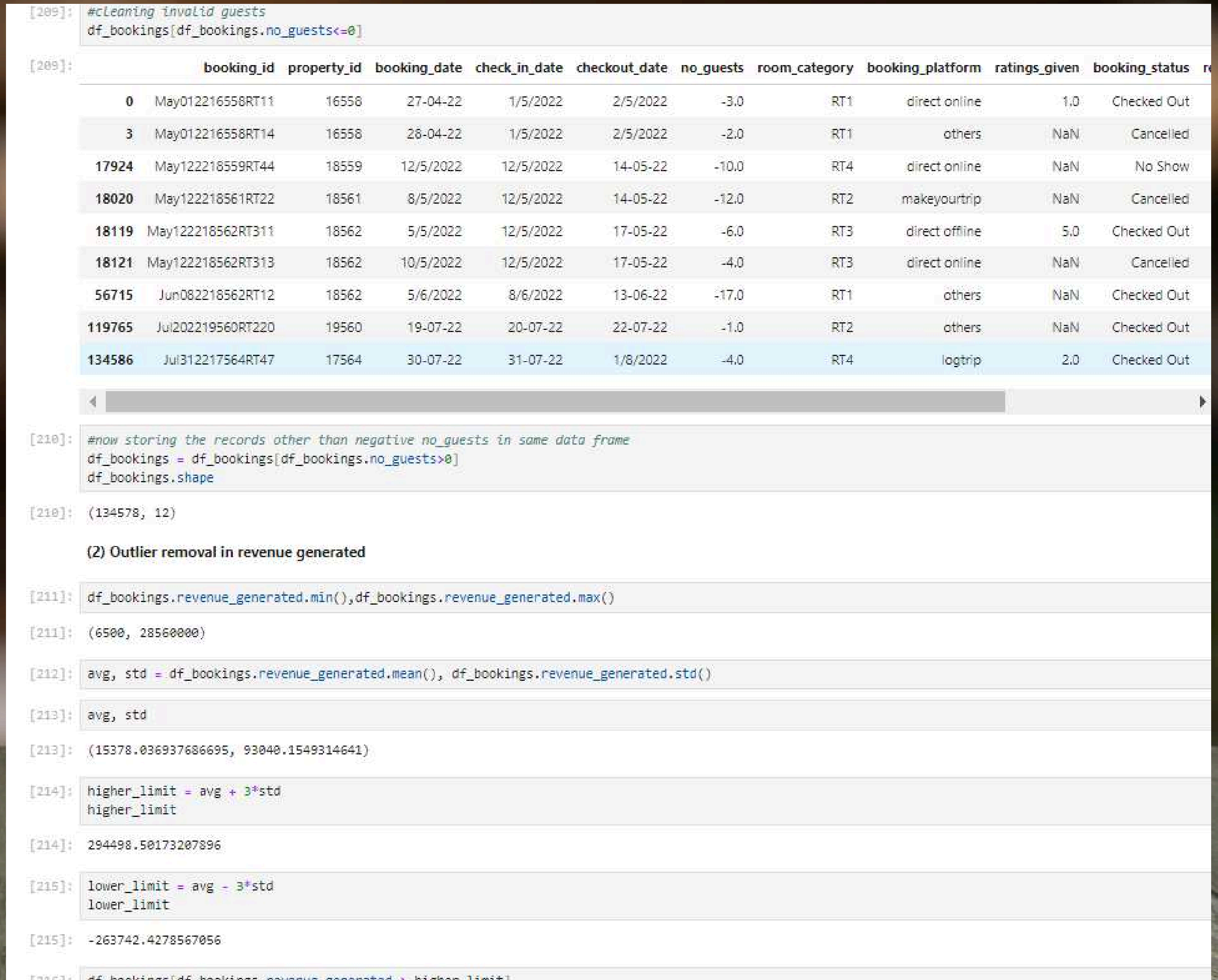
```
higher_limit = avg + 3*std
higher_limit
```

```
294498.50173207896
```

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

```
-263742.4278567056
```

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



```
[209]: #cleaning invalid guests
       df_bookings[df_bookings.no_guests<=0]
```

[209]:

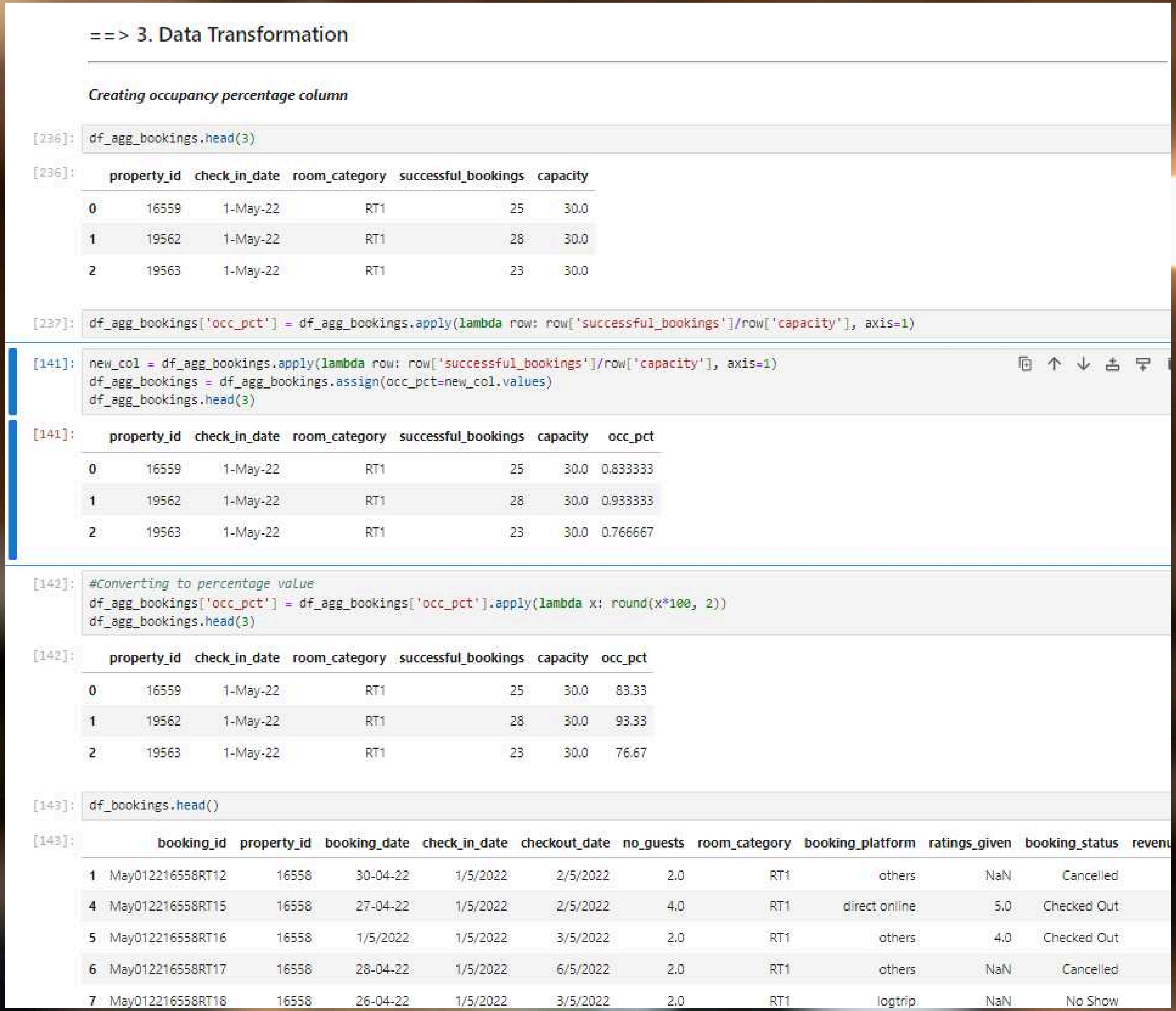
	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	r
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	1.0	Checked Out	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	NaN	Cancelled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

```
[210]: #now storing the records other than negative no_guests in same data frame
       df_bookings = df_bookings[df_bookings.no_guests>0]
       df_bookings.shape

[210]: (134578, 12)
```

(2) Outlier removal in revenue generated

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

[211]: (6500, 28560000)

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

[213]: avg, std

[213]: (15378.036937686695, 93040.1549314641)

[214]: higher_limit = avg + 3*std
       higher_limit

[214]: 294498.50173207896

[215]: lower_limit = avg - 3*std
       lower_limit

[215]: -263742.4278567056

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

DATA TRANSFORMATION



==> 3. Data Transformation

Creating occupancy percentage column

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

[236]:

	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

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

```
[141]: 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)
```

[141]:

	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

```
[142]: #Converting to percentage value
       df_agg_bookings['occ_pct'] = df_agg_bookings['occ_pct'].apply(lambda x: round(x*100, 2))
       df_agg_bookings.head(3)
```

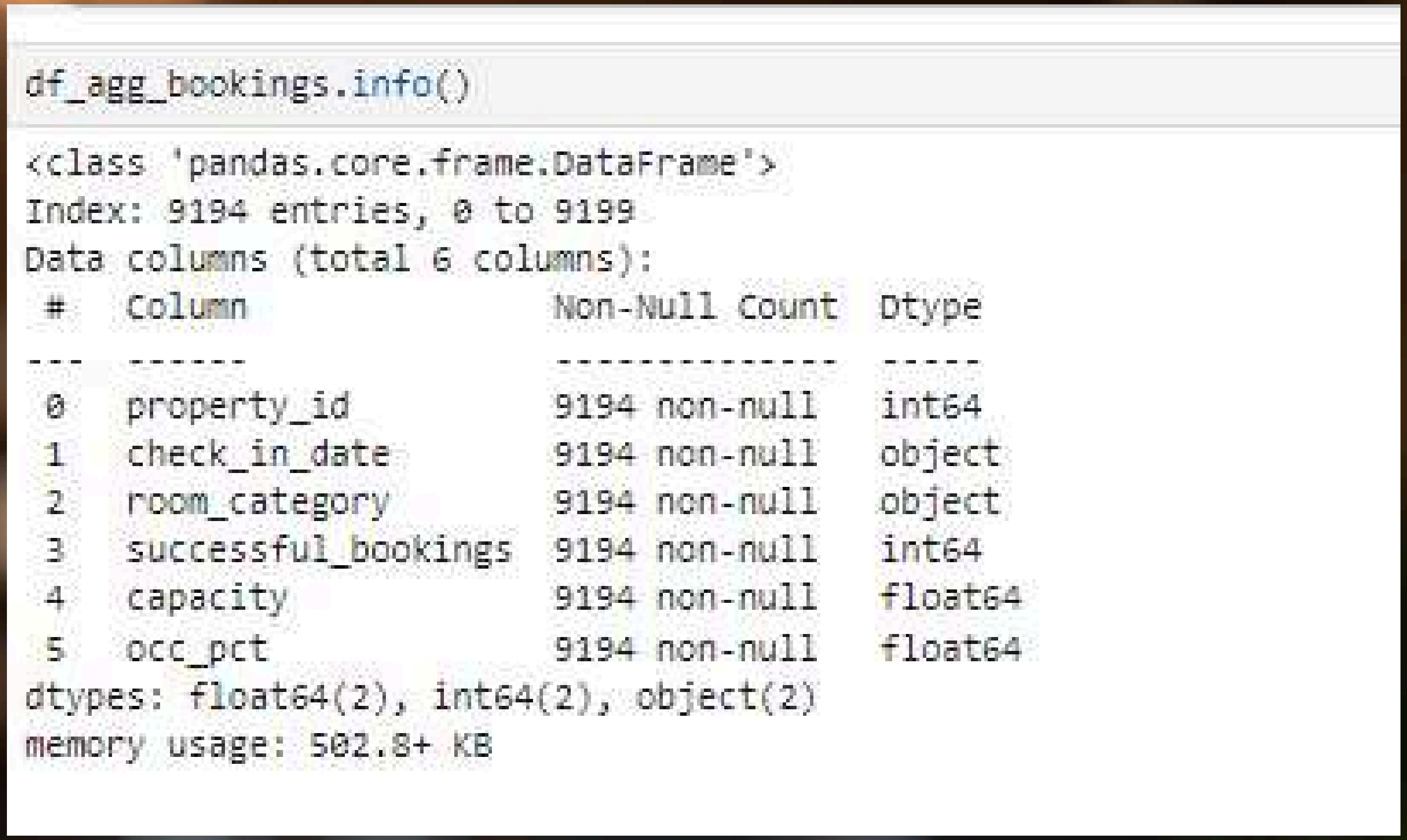
[142]:

	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

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

[143]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenu
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	

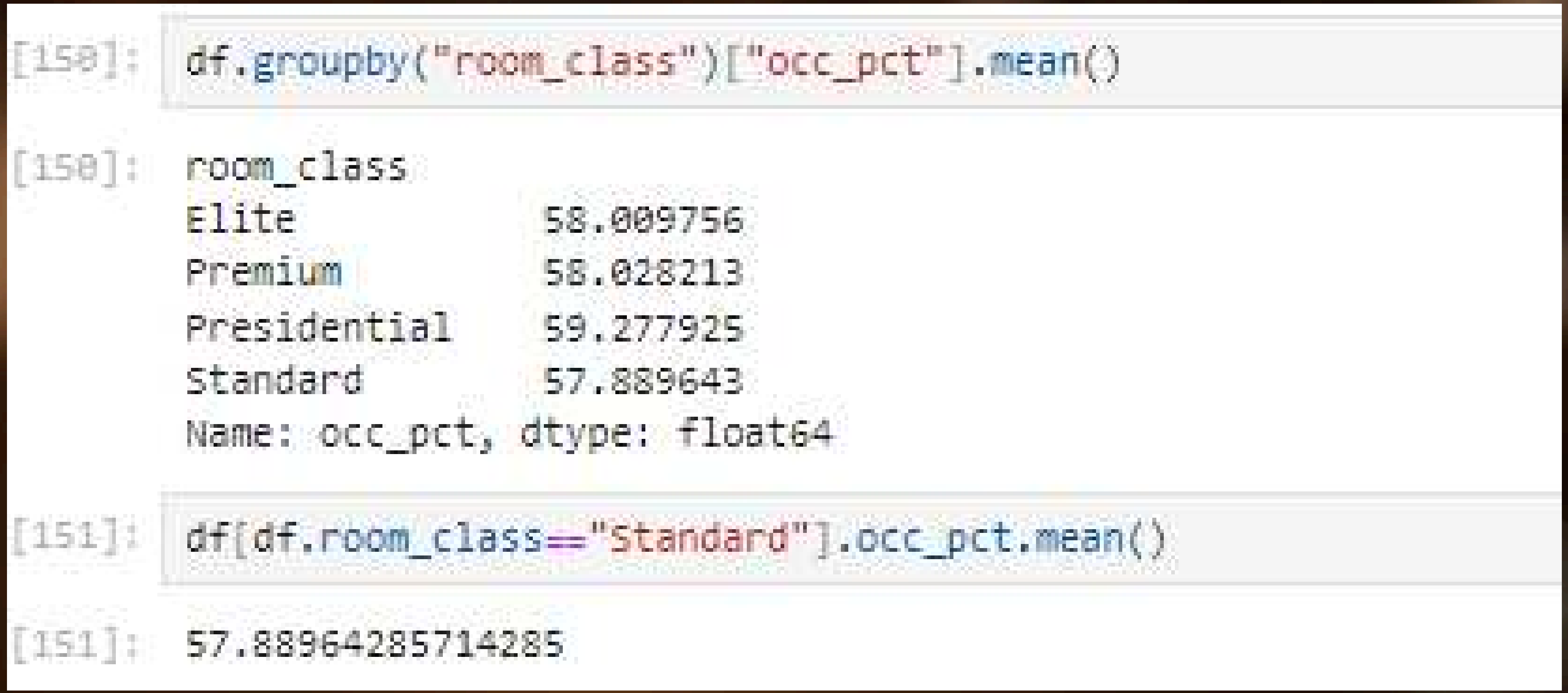


```
df_agg_bookings.info()
```

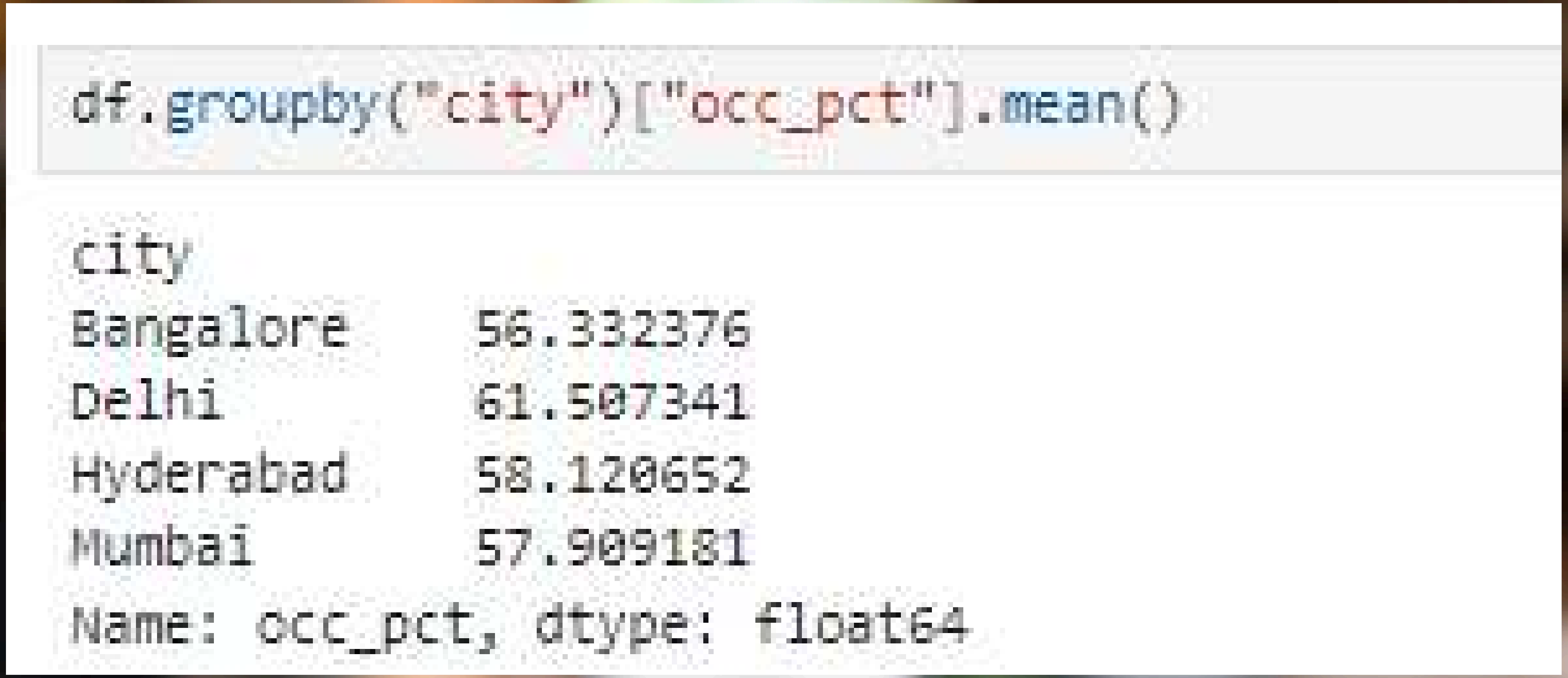
```
<class 'pandas.core.frame.DataFrame'>
Index: 9194 entries, 0 to 9199
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   property_id          9194 non-null   int64  
 1   check_in_date        9194 non-null   object 
 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
```

INSIGHTS GENERATION

Average Occupany Rate



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

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

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

[151]:  57.88964285714285
```

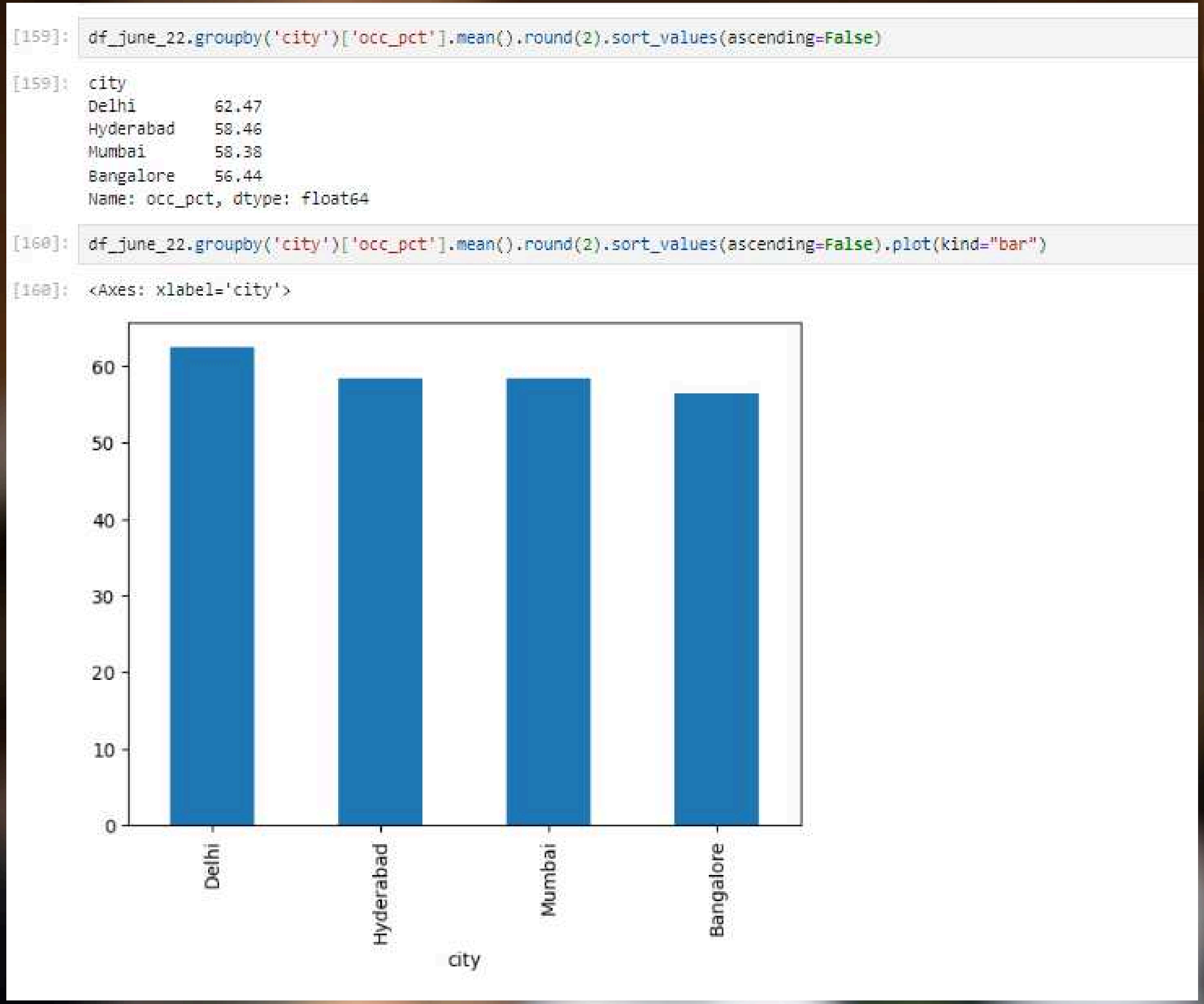


```
df.groupby("city")["occ_pct"].mean()

city
Bangalore    56.332376
Delhi        61.507341
Hyderabad    58.120652
Mumbai       57.909181
Name: occ_pct, dtype: float64
```

Average Occupany Rate per city

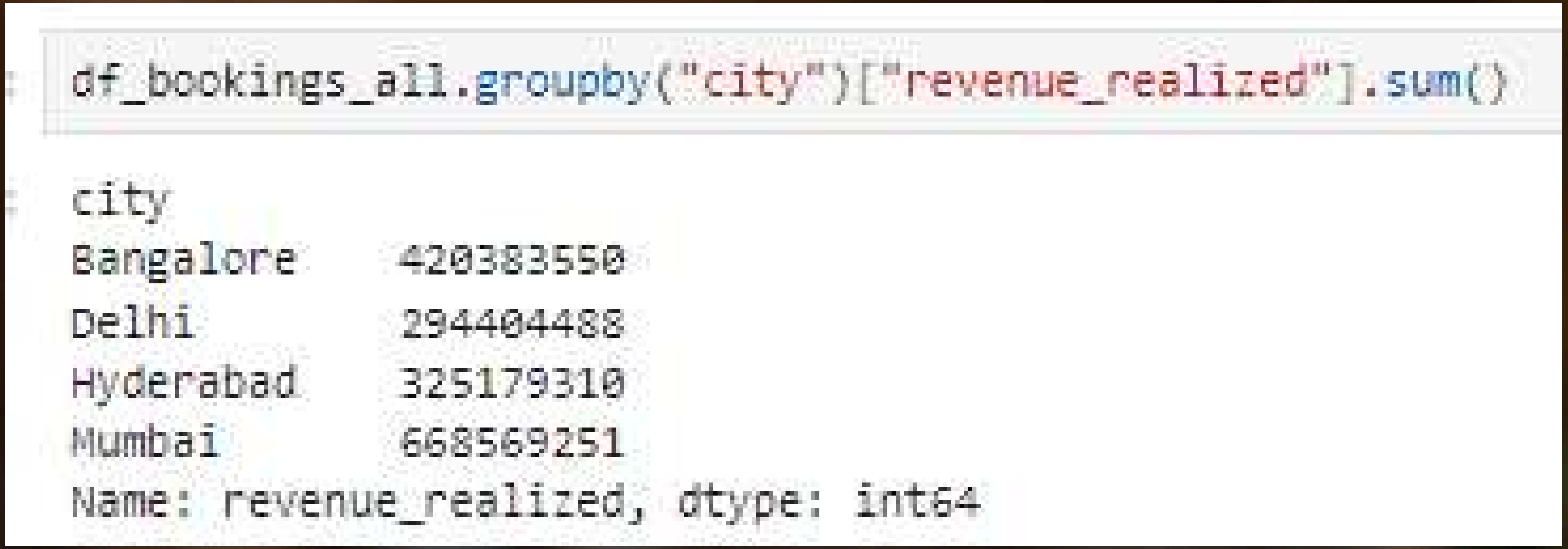
INSIGHTS GENERATION

Occupancy for different cities in month of June



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

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

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

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

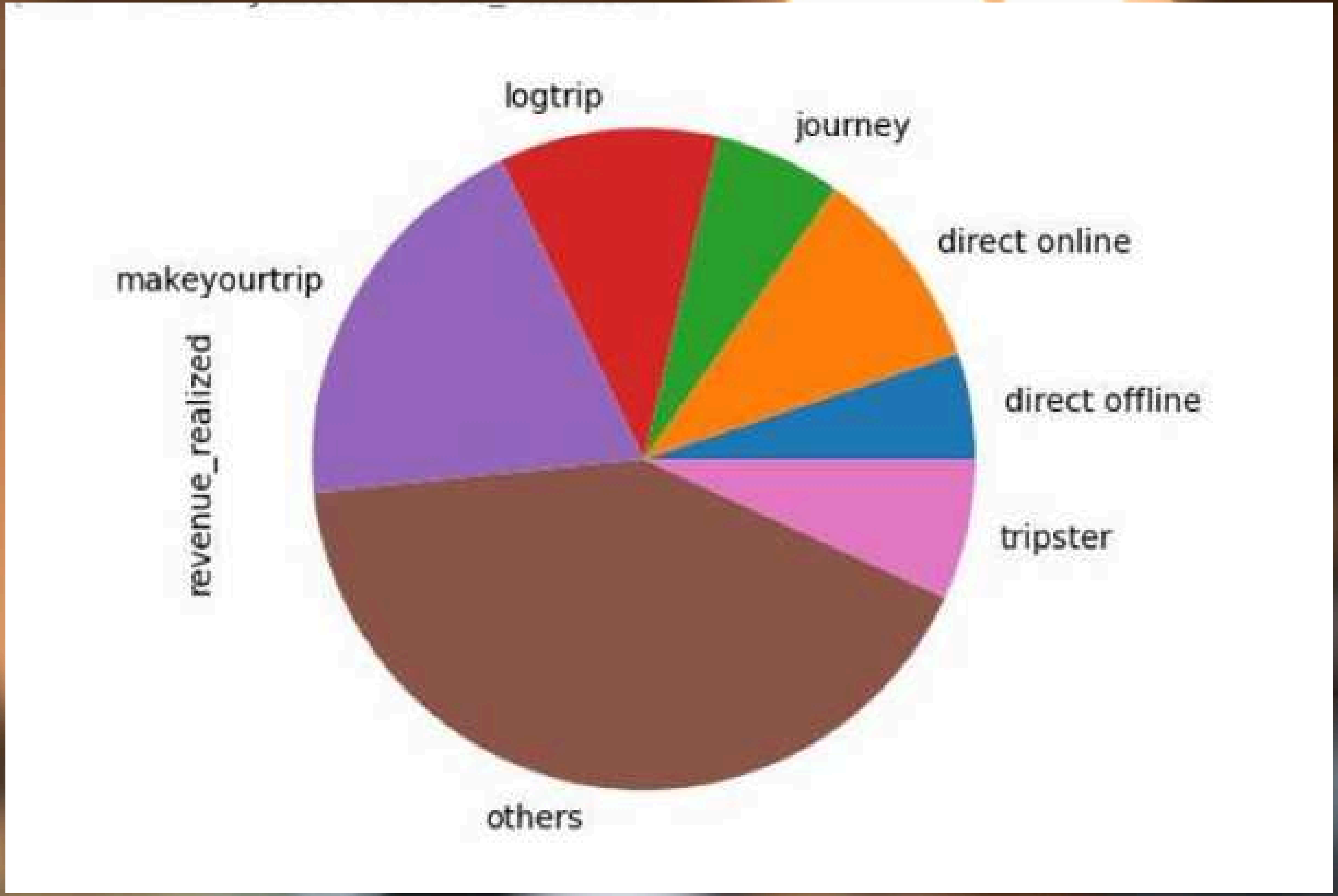
Revenue realized per city



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

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

INSIGHTS GENERATION

Revenue realized per booking platform



AtliQ GRANDS

THANK YOU!

For Showing Your interest

Presented by
Shimran Gupta