AtliQ

Hospitality Domain

Data Analysis Using Python

By Himanshu .K

Objective

To analyze historical booking and revenue data for Atliq Hospitality in order to uncover key performance insights, detect anomalies, and recommend data-driven strategies to enhance revenue generation and platform efficiency.

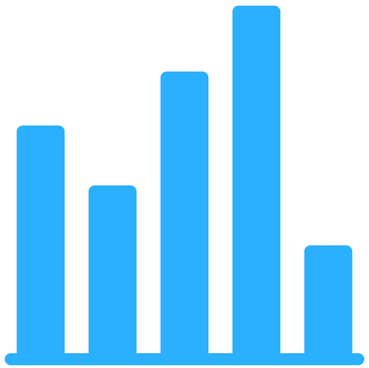
Problem Statement

Atliq Hospitality operates across various platforms but lacks clear insights into which platforms contribute the most revenue, how booking patterns affect realized revenue, and where potential revenue losses or anomalies may exist. There is a need to identify outliers, evaluate performance by platform, and recommend actionable strategies to optimize revenue streams.

Dataset Overview

1. fact_bookings.csv – Contains detailed booking-level data including booking_platform, revenue_generated, and revenue_realized.
2. dim_rooms.csv – Metadata about room types.
3. dim_hotels.csv – Metadata about hotel details.
4. dim_date.csv – Contains date-level granularity for time-based analysis.
5. fact_aggregated_bookings.csv – Aggregated summary of booking metrics.

Tools & Techniques Used
Python libraries: Pandas, Matplotlib, Seaborn, Plotly



Data Cleaning & Preparation

Importing Libraries into Python

```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import plotly.colors as  colors
pio.templates.default = 'plotly_white'
import seaborn as sns
```

```
df_bookings=pd.read_csv("C:/Users/91854/Desktop/fact_bookings.csv")
df_rooms =pd.read_csv("C:/Users/91854/Downloads/dim_rooms.csv")
df_hotels =pd.read_csv("C:/Users/91854/Downloads/dim_hotels.csv")
df_date =pd.read_csv("C:/Users/91854/Downloads/dim_date.csv")
df_agg_bookings =pd.read_csv("C:/Users/91854/Downloads/fact_aggregated_bookings.csv")
```

Exploring The Dataset

```
df_bookings
```

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revei
0	May012216558RT11	16558	2022-04-27	2022-05-01	2022-05-02	3	RT1	direct online	1.0	Checked Out	
1	May012216558RT12	16558	2022-04-30	2022-05-01	2022-05-02	2	RT1	others	NaN	Cancelled	
2	May012216558RT13	16558	2022-04-28	2022-05-01	2022-05-04	2	RT1	logtrip	5.0	Checked Out	
3	May012216558RT14	16558	2022-04-28	2022-05-01	2022-05-02	2	RT1	others	NaN	Cancelled	
4	May012216558RT15	16558	2022-04-27	2022-05-01	2022-05-02	4	RT1	direct online	5.0	Checked Out	
...	...	...	...	...	...	...	...	...	...	...	
134585	Jul312217564RT46	17564	2022-07-29	2022-07-31	2022-08-03	1	RT4	makeyourtrip	2.0	Checked Out	
134586	Jul312217564RT47	17564	2022-07-30	2022-07-31	2022-08-01	4	RT4	logtrip	2.0	Checked Out	
134587	Jul312217564RT48	17564	2022-07-30	2022-07-31	2022-08-02	1	RT4	tripster	NaN	Cancelled	
134588	Jul312217564RT49	17564	2022-07-29	2022-07-31	2022-08-01	2	RT4	logtrip	2.0	Checked Out	
134589	Jul312217564RT410	17564	2022-07-31	2022-07-31	2022-08-01	2	RT4	makeyourtrip	NaN	Cancelled	

134590 rows × 12 columns

DataFrame’s structure, including row count, column names, non-null counts, data types, and memory usage.

Your paragrap h text

```
df_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134590 entries, 0 to 134589
Data columns (total 12 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          134590 non-null  int64  
 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  
dtypes: float64(1), int64(4), object(7)
memory usage: 12.3+ MB
```

Generating descriptive statistics

```
df_bookings.describe()
```

	property_id	no_guests	ratings_given	revenue_generated	revenue_realized
count	134590.000000	134590.000000	56683.000000	134590.000000	134590.000000
mean	18061.113493	2.036808	3.619004	14916.013188	12696.123256
std	1093.055847	1.031766	1.235009	6452.868072	6928.108124
min	16558.000000	1.000000	1.000000	6500.000000	2600.000000
25%	17558.000000	1.000000	3.000000	9900.000000	7600.000000
50%	17564.000000	2.000000	4.000000	13500.000000	11700.000000
75%	18563.000000	2.000000	5.000000	18000.000000	15300.000000
max	19563.000000	6.000000	5.000000	45220.000000	45220.000000

Summary statistics of numeric columns in a DataFrame, including count, mean, standard deviation, min, max, and percentiles. It helps to understand the data better.

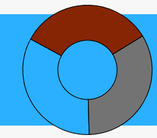
Checking for the null values in the dataset

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

```
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       77907
booking_status          0
revenue_generated       0
revenue_realized        0
dtype: int64
```

Since there are null values in the ratings column so I am not in need to remove those null values. .

Outlier Detection Using IQR Method



Your paragraph text

```
Q1 = df_bookings['revenue_generated'].quantile(0.25)
Q3 = df_bookings['revenue_generated'].quantile(0.75)
IQR =Q3-Q1
lowerbound= Q1 - 1.5*IQR
upperbound =Q3 + 1.5*IQR
outlier_rev= df_bookings[(df_bookings['revenue_generated']<lowerbound) |(df_bookings['revenue_generated']>upperbound)]
outlier_rev.shape[0]
```

```
6284
```

```
df_bookings['rev_outlier_flag'] = df_bookings['revenue_generated'].apply(
    lambda x: 'Outlier' if x < lowerbound or x > upperbound else 'Normal'
)
df_bookings
```

te	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_generated	revenue_realized	revenue_difference	rev_outlier_flag
01	2022-05-02	3	RT1	direct online	1.0	Checked Out	10010	10010	0	Normal
01	2022-05-02	2	RT1	others	NaN	Cancelled	9100	3640	5460	Normal
01	2022-05-04	2	RT1	logtrip	5.0	Checked Out	9100	9100	0	Normal
01	2022-05-02	2	RT1	others	NaN	Cancelled	9100	3640	5460	Normal
01	2022-05-02	4	RT1	direct online	5.0	Checked Out	10920	10920	0	Normal
...	...	...	...	...	...	...	...	...	...	...
31	2022-08-03	1	RT4	makeyourtrip	2.0	Checked Out	32300	32300	0	Outlier
31	2022-08-01	4	RT4	logtrip	2.0	Checked Out	38760	38760	0	Outlier
31	2022-08-02	1	RT4	tripster	NaN	Cancelled	32300	12920	19380	Outlier
31	2022-08-01	2	RT4	logtrip	2.0	Checked Out	32300	32300	0	Outlier
31	2022-08-01	2	RT4	makeyourtrip	NaN	Cancelled	32300	12920	19380	Outlier

Since the no. of outliers is significantly more , I will flag the outliers instead of removing it because it might affect the analysis.

Revenue Generated Through the Booking Platform

```
revenue_by_platform= df_bookings.groupby('booking_platform')['revenue_generated'].sum().reset_index()
revenue_by_platform
```

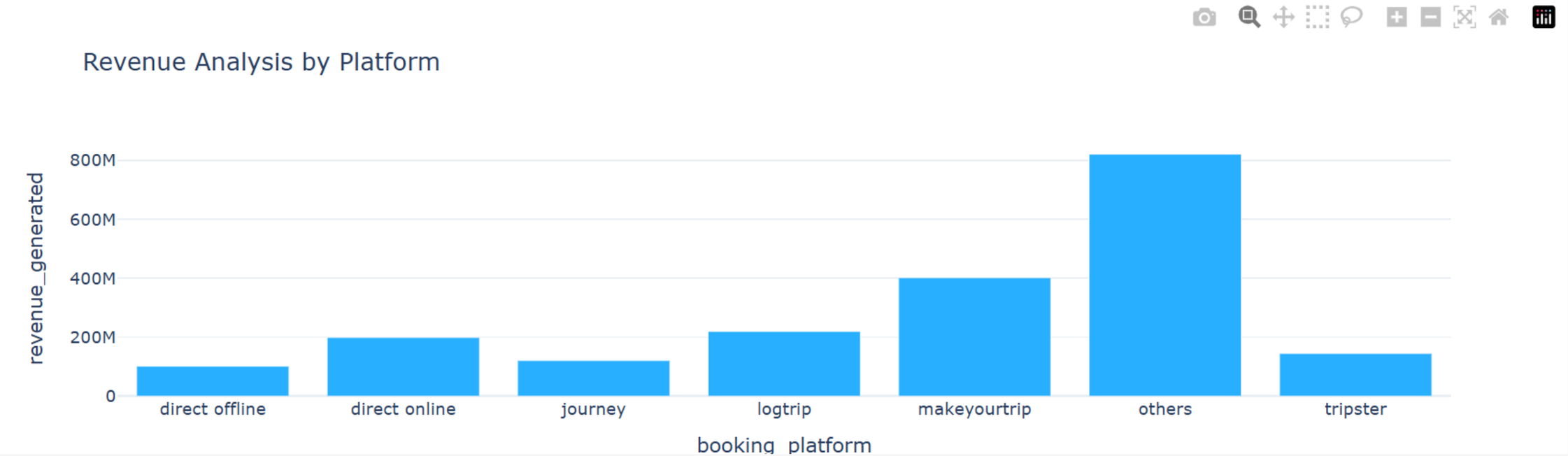
	booking_platform	revenue_generated
0	direct offline	101359255
1	direct online	198770030
2	journey	120654715
3	logtrip	219478955
4	makeyourtrip	401502130
5	others	821066620
6	tripster	144714510

Generated Revenue: This typically refers to the total revenue produced or expected to be produced, including both realized and unrealized portions.

Revenue Generated by Booking Platform — Visualized Using a Bar Graph

```
fig1 = px.bar(revenue_by_platform,
              x='booking_platform',
              y='revenue_generated',
              title ='Revenue Analysis by Platform',
                          color_discrete_sequence=['#2ab0ff'] )

fig1.show()
```



Insight:Revenue generated from other platforms is high followed by makeyourtrip and .logtrip

Realized Revenue by Each Platform

```
realrevenue_by_platform= df_bookings.groupby('booking_platform')['revenue_realized'].sum().reset_index()
realrevenue_by_platform
```

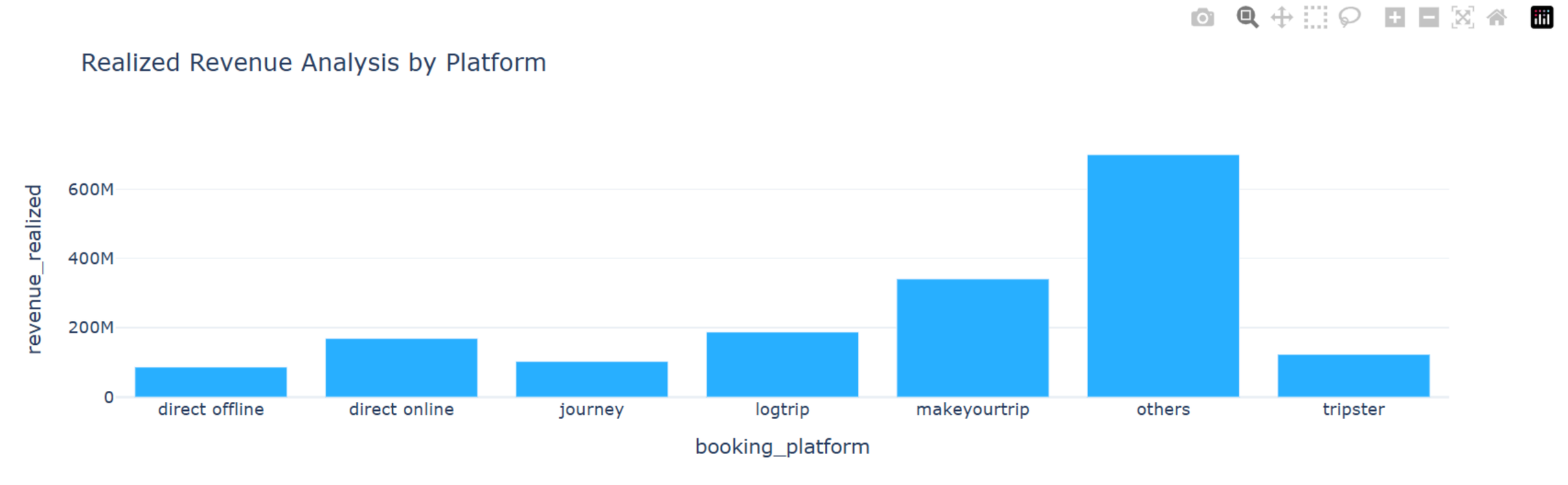
	booking_platform	revenue_realized
0	direct offline	86404333
1	direct online	169026467
2	journey	102531334
3	logtrip	187554488
4	makeyourtrip	340834504
5	others	699353302
6	tripster	123066801

Realized Revenue:This refers to revenue that has actually been earned and recognized in the books. It's based on completed sales transactions or milestones reached.

Revenue Realized by Booking Platform — Visualized Using a Bar Graph

```
figure = px.bar(realrevenue_by_platform,
             x='booking_platform',
             y='revenue_realized',
             title ='Realized Revenue Analysis by Platform',
                         color_discrete_sequence=['#2ab0ff'] )

figure.show()
```



Insight:Revenue realized from other platforms is high but makeyourtrip and logtrip has also done good.

.

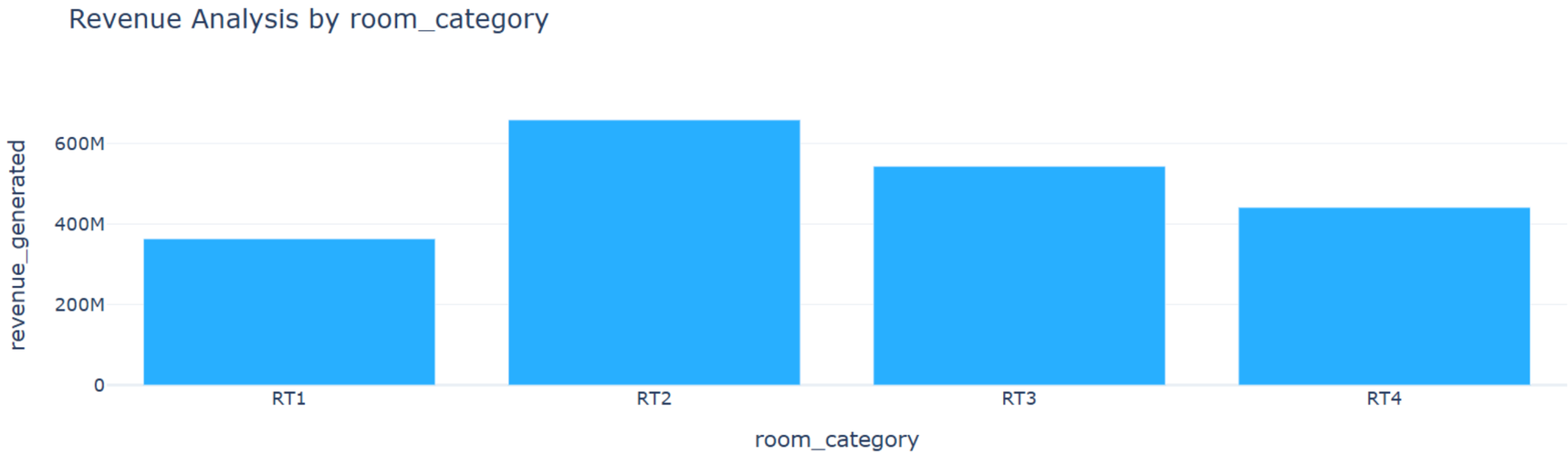
Revenue Analysis by Room - Category

```
revenue_by_room_category= df_bookings.groupby('room_category')['revenue_generated'].sum().reset_index()
revenue_by_room_category
```

	room_category	revenue_generated
0	RT1	363545195
1	RT2	658946160
2	RT3	543597840
3	RT4	441457020

```
fig2 = px.bar(revenue_by_room_category,
              x='room_category',
              y='revenue_generated',
              title ='Revenue Analysis by room_category',
                          color_discrete_sequence=['#2ab0ff'] )

fig2.show()
```



Insight: As the bar-graph shows that the RT2 room has generated the maximum revenue followed by RT3 and RT4.

Table Creation By Using Merge Function

```
df_hotels_bookings =pd.merge(df_bookings, df_hotels, on ='property_id')
df_hotels_bookings
```

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	reve
0	May012216558RT11	16558	2022-04-27	2022-05-01	2022-05-02	3	RT1	direct online	1.0	Checked Out	
1	May012216558RT12	16558	2022-04-30	2022-05-01	2022-05-02	2	RT1	others	NaN	Cancelled	
2	May012216558RT13	16558	2022-04-28	2022-05-01	2022-05-04	2	RT1	logtrip	5.0	Checked Out	
3	May012216558RT14	16558	2022-04-28	2022-05-01	2022-05-02	2	RT1	others	NaN	Cancelled	
4	May012216558RT15	16558	2022-04-27	2022-05-01	2022-05-02	4	RT1	direct online	5.0	Checked Out	
...	...	...	...	...	...	...	...	...	...	...	
134585	Jul312217564RT46	17564	2022-07-29	2022-07-31	2022-08-03	1	RT4	makeyourtrip	2.0	Checked Out	
134586	Jul312217564RT47	17564	2022-07-30	2022-07-31	2022-08-01	4	RT4	logtrip	2.0	Checked Out	
134587	Jul312217564RT48	17564	2022-07-30	2022-07-31	2022-08-02	1	RT4	tripster	NaN	Cancelled	
134588	Jul312217564RT49	17564	2022-07-29	2022-07-31	2022-08-01	2	RT4	logtrip	2.0	Checked Out	
134589	Jul312217564RT410	17564	2022-07-31	2022-07-31	2022-08-01	2	RT4	makeyourtrip	NaN	Cancelled	

134590 rows × 15 columns

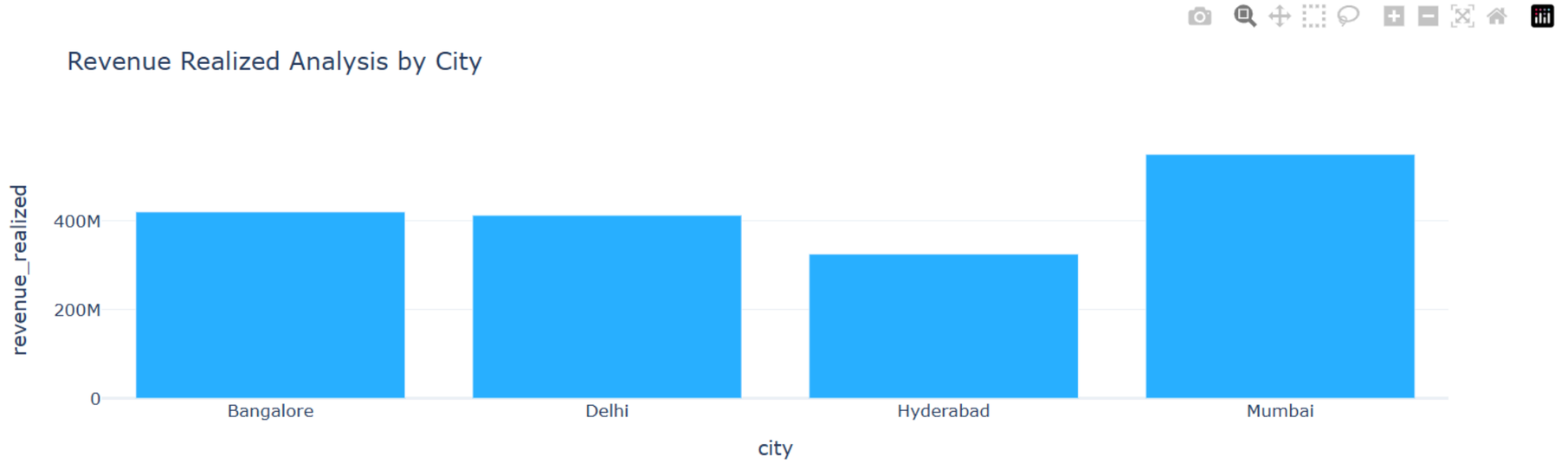
Created a new dataframe named hotel bookings by merging bookings and hotels dataframe for further analysis.

Realized Revenue Analysis by City

```
revenue_realized_by_city = df_hotels_bookings.groupby('city')['revenue_realized'].sum().reset_index()
revenue_realized_by_city
```

	city	revenue_realized
0	Bangalore	420397050
1	Delhi	412948736
2	Hyderabad	325232870
3	Mumbai	550192573

```
fig2 =px.bar(revenue_realized_by_city,
             x = 'city',
             y = 'revenue_realized',
             title ='Revenue Realized Analysis by City',
              color_discrete_sequence=['#2ab0ff'] )
fig2.show()
```



Insight: Mumbai has contributed the highest in the revenue realized while Hyderabad has the least contribution.

.

Table Creation By Using Merge Function

```
df_occup= pd.merge(df_agg_bookings,df_rooms, left_on ='room_category', right_on = 'room_id')
df_occup
```

	property_id	check_in_date	room_category	successful_bookings	capacity	room_id	room_class
0	16559	01-May-22	RT1	25	30	RT1	Standard
1	19562	01-May-22	RT1	28	30	RT1	Standard
2	19563	01-May-22	RT1	23	30	RT1	Standard
3	17558	01-May-22	RT1	13	19	RT1	Standard
4	16558	01-May-22	RT1	18	19	RT1	Standard
...	...	...	...	...	...	...	...
9195	16563	31-Jul-22	RT4	13	18	RT4	Presidential
9196	16559	31-Jul-22	RT4	13	18	RT4	Presidential
9197	17558	31-Jul-22	RT4	3	6	RT4	Presidential
9198	19563	31-Jul-22	RT4	3	6	RT4	Presidential
9199	17561	31-Jul-22	RT4	3	4	RT4	Presidential

9200 rows × 7 columns

Created a new dataframe named occup by merging aggregated bookings and rooms dataframe for further analysis.

.

Successful Bookings Analysis By Room Class

```
Successful_bookings = df_occup.groupby('room_class')['successful_bookings'].sum().reset_index()
Successful_bookings
```

	room_class	successful_bookings
0	Elite	49505
1	Premium	30566
2	Presidential	16073
3	Standard	38446

Insight: Elite Rooms has the highest number of successful bookings among all the room class while Standard Rooms have second highest successful bookings..

.

Room Class With Capacity

```
capacity = df_occup.groupby('room_class')['capacity'].sum().reset_index()
capacity
```

	room_class	capacity
0	Elite	85928
1	Premium	53084
2	Presidential	27140
3	Standard	66424

```
df_occupied= pd.merge(Successful_bookings,capacity, on = 'room_class')
df_occupied
```

	room_class	successful_bookings	capacity
0	Elite	49505	85928
1	Premium	30566	53084
2	Presidential	16073	27140
3	Standard	38446	66424

Successful Bookings By Percentage

```
df_occupied['percentage_successful_bookings'] = (
    df_occupied['successful_bookings'] / df_occupied['capacity']
) * 100
df_occupied['percentage_successful_bookings'] = df_occupied['percentage_successful_bookings'].round(2)
df_occupied
```

	room_class	successful_bookings	capacity	percentage_successful_bookings
0	Elite	49505	85928	57.61
1	Premium	30566	53084	57.58
2	Presidential	16073	27140	59.22
3	Standard	38446	66424	57.88

Successful Bookings Analysis By Percentage

```
fig = px.bar(df_occupied,
             x='percentage_successful_bookings',
             y='room_class',
             orientation='h',
             title='Successful Booking Percentage by Room Class',
             text='percentage_successful_bookings',
             color='room_class',  # Adds color variation
             color_discrete_sequence=['#2ab0ff'])
fig.update_traces(
    texttemplate='%{text:.2f}%',
    textposition='outside')
fig.show()
```

Insight: The Presidential room class leads in successful bookings at 59.22%, suggesting it's the most reliably reserved among all room types.

Realized Revenue by Property

```
revenue_realizedperhotel_type = df_property.groupby('property_name')['revenue_realized'].sum().reset_index()
revenue_realizedperhotel_type
```

	property_name	revenue_realized
0	Atliq Bay	260051178
1	Atliq Blu	260855522
2	Atliq City	285811939
3	Atliq Exotica	320312468
4	Atliq Grands	211532764
5	Atliq Palace	304081863
6	Atliq Seasons	66125495

Realized Revenue Analysis by Property

```
figg = px.bar(revenue_realizedperhotel_type,
              x='property_name',
              y= 'revenue_realized',
              title='Realized Revenue By Property',
              color_discrete_sequence=['#2ab0ff'] )
figg.show()
```

Insight: Atliq Exotica has the highest realized revenue among all the properties shows that it one of the most popular among all .

🔍 Key Insights:

Revenue Distribution by Platform: Significant differences in revenue-generated across different booking platforms.A clear leader in platform performance is identified using grouped bar charts.

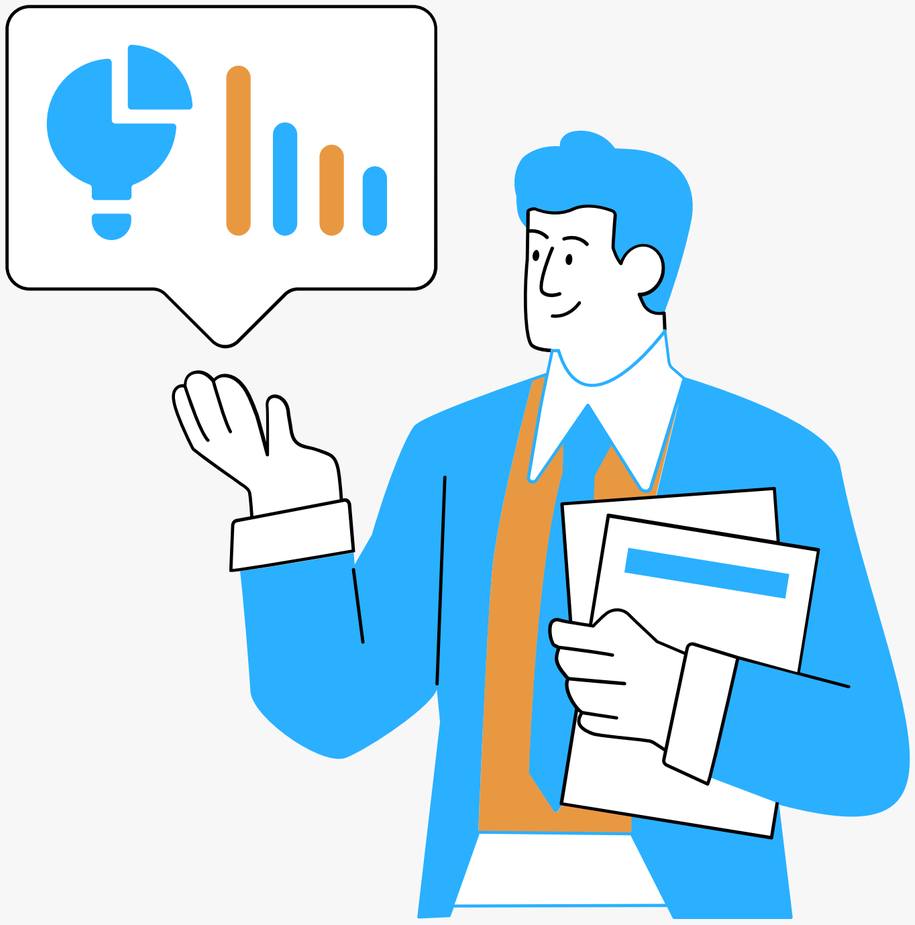
Outlier Detection :Used IQR method to detect revenue anomalies. Several outliers found in the revenue-generated column which might indicate abnormal bookings or data entry errors.

Realized Revenue vs Generated Revenue: Some platforms have a noticeable gap between revenue generated and realized revenue, indicating leakages or failed transactions.

Visualizations: Effective use of Plotly charts to present revenue by platform. Side-by-side comparison of metrics helps stakeholders see gaps and opportunities.

Course Of Action

1. Optimize High-Performing Platforms: Focus marketing and partnerships on platforms yielding higher realized revenue.
2. Investigate Revenue Outliers: Perform root cause analysis on high and low outlier bookings to identify fraud or system issues.
3. Improve Conversion Rate: For platforms with high generated but low realized revenue, identify friction points in the booking process.
4. Enhance Data Quality Checks: Introduce automated outlier detection and data validation mechanisms for financial metrics.
5. Integrate Time-Series Trends: Use the dim_date file to extend the analysis into seasonality and trends across months or quarters for forecasting.



Thank You

Project by Himanshu.K