Ad Hoc ANALYSIS

HOSPITALITY DOMAIN

Table of CONTENTS

INTRODUCTION

AtliQ Grands, a prominent name in Indian hospitality for over two decades, faced challenges with declining market share and revenue in their luxury/business accommodations due to intense competition.

Our goal was to help AtliQ Grands reclaim their market share and revenue by integrating "Business and Data Intelligence" into their operations.

EXPLORATORY DATA ANALYSIS

Importing Pandas library

In [1]:
```
1 import pandas as pd
```

Reading data files

In [2]:
```
1 df_bookings = pd.read_csv('datasets//fact_bookings.csv')
2 df_hotels = pd.read_csv('datasets//dim_hotels.csv')
3 df_rooms = pd.read_csv('datasets//dim_rooms.csv')
4 df_agg_bookings = pd.read_csv('datasets//fact_aggregated_bookings.csv')
5 df_date = pd.read_csv('datasets//dim_date.csv')
```

Exploratory Data Analysis: fact_bookings data

In [3]:
```
1 df_bookings.head()
```

Out[3]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	1.0
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	5.0
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	NaN
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4.0	RT1	direct online	5.0

In [4]:
```
1 df_bookings.shape # Gives dimensions
```

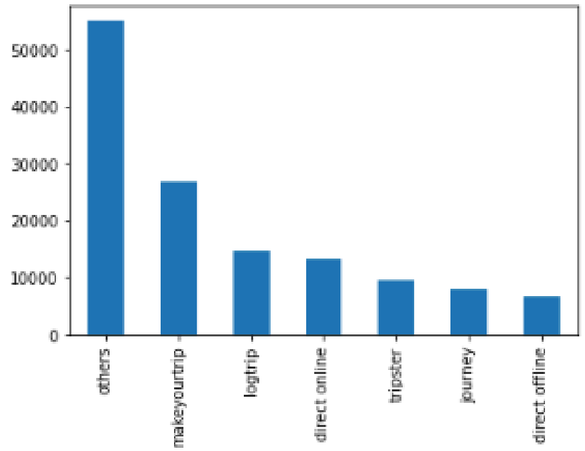
Out[4]: (134590, 12)

In [5]:
```
1 df_bookings.describe() # Gives quick statistics on Numerical columns
```

EXPLORATORY DATA ANALYSIS

```
In [9]:  1  df_bookings.booking_platform.value_counts().plot(kind="bar")  # Bar plot
```

Out[9]: <AxesSubplot:>



Exploratory Data Analysis: dim_hotels data

```
In [11]:  1  df_hotels.head()
```

Out[11]:

	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 [12]:  1  df_hotels.shape
```

Out[12]: (25, 4)

```
In [13]:  1  df_hotels.category.value_counts()
```

Out[13]:
```
Luxury      16
Business     9
Name: category, dtype: int64
```

```
In [14]:  1  df_hotels.city.value_counts().sort_values()
```

Out[14]:
```
Delhi        5
Hyderabad    6
Bangalore    6
Mumbai       8
Name: city, dtype: int64
```

EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis: fact_aggregate_bookings data

In [16]:
```
1 df_agg_bookings.head()
```

Out[16]:

	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 [17]:
```
1 df_agg_bookings.property_id.unique()
```

Out[17]:
```
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], dtype=int64)
```

In [18]:
```
1 df_agg_bookings.groupby('property_id')['successful_bookings'].sum()
```

Out[18]:
```
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
```

In [20]:
```
1 df_agg_bookings.describe()
```

Out[20]:

	property_id	successful_bookings	capacity
count	9200.000000	9200.000000	9198.000000
mean	18040.640000	14.655761	25.280496
std	1099.818325	7.736170	11.442080
min	16558.000000	1.000000	3.000000
25%	17558.000000	9.000000	18.000000
50%	17564.000000	14.000000	25.000000
75%	18563.000000	19.000000	34.000000
max	19563.000000	123.000000	50.000000

In [21]:
```
1 df_agg_bookings.room_category.value_counts()
```

Out[21]:
```
RT1    2300
RT2    2300
RT3    2300
RT4    2300
Name: room_category, dtype: int64
```

In [22]:
```
1 # Days on which bookings are greater than capacity
2 df_agg_bookings[df_agg_bookings.successful_bookings > df_agg_bookings.capacity]
```

DATA CLEANING

Data Cleaning

In [26]:
```
1 df_bookings.describe()
```

Out[26]:

	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

In [27]:
```
1 df_bookings[df_bookings.no_guests < 0]
```

In [31]:
```
1 avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.std()
```

In [32]:
```
1 avg, std
```

Out[32]: (15378.036937686695, 93040.15493143328)

In [33]:
```
1 higher_lim = avg + 3*std
2 higher_lim
```

Out[33]: 294498.50173198653

In [34]:
```
1 lower_lim = avg - 3*std
2 lower_lim
```

Out[34]: -263742.4278566132

In [35]:
```
1 df_bookings[df_bookings.revenue_generated > higher_lim] # Shows the outliers
```

DATA CLEANING

In [39]:
```
1 df_bookings.revenue_realized.describe()
```

Out[39]:
```
count    134573.000000
mean      12695.983585
std        6927.791692
min        2600.000000
25%        7600.000000
50%       11700.000000
75%       15300.000000
max       45220.000000
Name: revenue_realized, dtype: float64
```

In [40]:
```
1 # Checking if the max value in revenue_realized column is valid or not
2 higher_lim = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std()
3 higher_lim
```

Out[40]:
```
33479.3586618449
```

In [41]:
```
1 df_bookings[df_bookings.revenue_realized > higher_lim]
```

Out[41]:

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

DATA TRANSFORMATION

Data Transformation

In [54]:
```
1 df_agg_bookings.head()
```

Out[54]:

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

Business terminology:

Occupancy percentage = Successful bookings / Capacity

In [55]:
```
1 # Add a column named occupancy ratio to the dataframe object
2 df_agg_bookings['occ_pct'] = df_agg_bookings['successful_bookings']/df_agg_bookings['capacity']
3 df_agg_bookings.head()
```

Out[55]:

	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
4	16558	1-May-22	RT1	18	19.0	0.947368
5	17560	1-May-22	RT1	28	40.0	0.700000

DATA TRANSFORMATION

In [56]:
```
df_agg_bookings
```

Out[56]:

	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
4	16558	1-May-22	RT1	18	19.0	0.947368
5	17560	1-May-22	RT1	28	40.0	0.700000
...	...	...	...	...	...	...
9195	16563	31-Jul-22	RT4	13	18.0	0.722222
9196	16559	31-Jul-22	RT4	13	18.0	0.722222
9197	17558	31-Jul-22	RT4	3	6.0	0.500000
9198	19563	31-Jul-22	RT4	3	6.0	0.500000
9199	17561	31-Jul-22	RT4	3	4.0	0.750000

9194 rows × 6 columns

In [57]:
```
df_agg_bookings['occ_pct'] = df_agg_bookings['occ_pct'].apply(lambda x: round(x/100, 2))
```

In [58]:
```
df_agg_bookings.head()
```

Out[58]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
0	16559	1-May-22	RT1	25	30.0	0.01
1	19562	1-May-22	RT1	28	30.0	0.01
2	19563	1-May-22	RT1	23	30.0	0.01
4	16558	1-May-22	RT1	18	19.0	0.01

INSIGHTS (AD HOC ANALYSIS)

1. Average occupancy rate in each of the room categories

In [59]:
```
1 df_agg_bookings.groupby('room_category')['occ_pct'].mean().round(2)
```

Out[59]:
```
room_category
RT1    0.01
RT2    0.01
RT3    0.01
RT4    0.01
Name: occ_pct, dtype: float64
```

In [60]:
```
1 df_rooms
```

Out[60]:

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

In [61]:
```
1 df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_on="room_id")
2 df.head()
```

Out[61]:

In [62]:
```
1 df.groupby('room_class')['occ_pct'].mean().round(2)
```

Out[62]:
```
room_class
Elite           0.01
Premium         0.01
Presidential    0.01
Standard        0.01
Name: occ_pct, dtype: float64
```

In [63]:
```
1 df.drop('room_id', axis=1, inplace=True)
```

INSIGHTS (AD HOC ANALYSIS)

2. Print average occupancy rate per city

In [65]:
```
1 df_hotels.head()
```

Out[65]:

	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

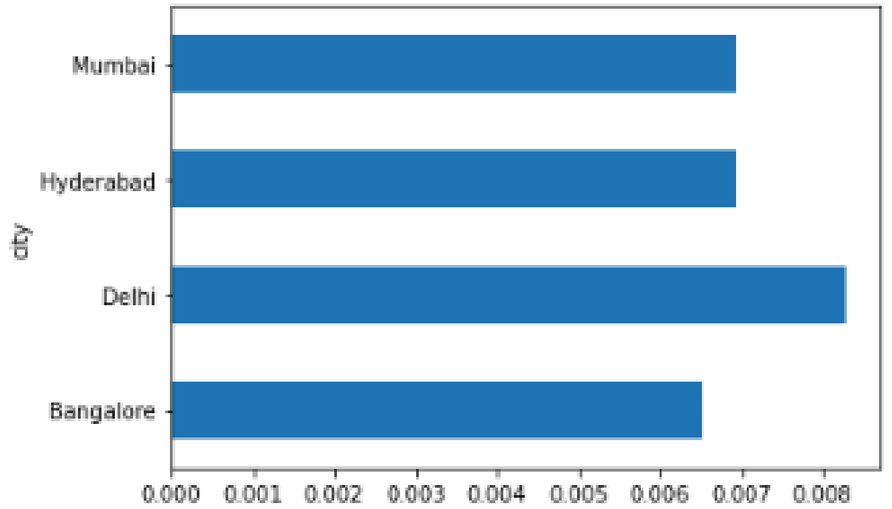
In [66]:
```
1 df = pd.merge(df, df_hotels, on='property_id')
2 df
```

In [68]:
```
1 df.groupby('city')['occ_pct'].mean()
```

Out[68]:
```
city
Bangalore    0.006522
Delhi        0.008282
Hyderabad    0.006937
Mumbai       0.006942
Name: occ_pct, dtype: float64
```

In [69]:
```
1 df.groupby('city')['occ_pct'].mean().plot(kind='barh')
```

Out[69]: <AxesSubplot:ylabel='city'>



INSIGHTS (AD HOC ANALYSIS)

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

```
In [70]:
1 df_date = pd.read_csv('datasets//dim_date.csv')
```

```
In [71]:
1 df_date.head()
```

```
In [75]:
1 df.head()
```

Out[75]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	mmm y
0	16559	10-May-22	RT1	18	30.0	0.01	Standard	Atliq Exotica	Luxury	Mumbai	Ma 2
1	16559	10-May-22	RT2	25	41.0	0.01	Elite	Atliq Exotica	Luxury	Mumbai	Ma 2
2	16559	10-May-22	RT3	20	32.0	0.01	Premium	Atliq Exotica	Luxury	Mumbai	Ma 2
3	16559	10-May-22	RT4	13	18.0	0.01	Presidential	Atliq Exotica	Luxury	Mumbai	Ma 2
4	19562	10-May-22	RT1	18	30.0	0.01	Standard	Atliq Bay	Luxury	Bangalore	Ma 2

```
In [76]:
1 df.groupby('day_type')['occ_pct'].mean().round(2)
```

```
Out[76]: day_type
weekeday    0.01
weekend     0.01
Name: occ_pct, dtype: float64
```

INSIGHTS (AD HOC ANALYSIS)

4. In the month of June, what is the occupancy rate of different cities

In [77]:
```
df_june_22 = df[df["mmm yy"]=='Jun 22'].groupby("city")["occ_pct"].mean().round(2)
```

In [78]:
```
df_june_22
```

Out[78]:
```
city
Bangalore    0.01
Delhi        0.01
Hyderabad    0.01
Mumbai       0.01
Name: occ_pct, dtype: float64
```

5. Incorporate new data: August data, and analyze

In [79]:
```
df_august = pd.read_csv("datasets//new_data_august.csv")
df_august.head()
```

Out[79]:

	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.
1	19562	Atliq Bay	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	21	30	70.
2	19563	Atliq Palace	Business	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	23	30	76.
3	19558	Atliq Grands	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	30	40	75.
4	19560	Atliq City	Business	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	20	26	76.

INSIGHTS (AD HOC ANALYSIS)

6. Print revenue realized per city

In [84]:
```
1 df_bookings.head()
```

Out[84]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status
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

In [85]:
```
1 df_bookings = pd.merge(df_bookings, df_hotels, on='property_id')
```

In [86]:
```
1 df_bookings.head()
```

Out[86]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status
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
3	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2.0	RT1	others	NaN	Cancelled
4	May012216558RT18	16558	26-04-22	1/5/2022	3/5/2022	2.0	RT1	logtrip	NaN	No Show

INSIGHTS (AD HOC ANALYSIS)

```
In [87]:
1 df_bookings.groupby("city")["revenue_realized"].sum().sort_values(ascending=False)
```

```
Out[87]: city
Mumbai       668569251
Bangalore    420383550
Hyderabad    325179310
Delhi        294404488
Name: revenue_realized, dtype: int64
```

7. Print month by month revenue

```
In [88]:
1 df_date_bookings = pd.merge(df_bookings, df_date, left_on="check_in_date", right_on="date")
```

```
In [89]:
1 df_date_bookings.head()
```

Out[89]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_statu

```
In [90]:
1 # no results shown above, because: the date variable is not stored with date type. It is stored with object data t
2 df_bookings.info(), df_date.info()
```

INSIGHTS (AD HOC ANALYSIS)

```
In [95]:  1  df_date_bookings.groupby('mmm yy')['revenue_realized'].sum().round(2)

Out[95]: mmm yy
         Jul 22    389940912
         Jun 22    377191229
         May 22    408375641
         Name: revenue_realized, dtype: int64
```

```
In [98]:  1  df_bookings.groupby('category')["revenue_realized"].sum()

Out[98]: category
         Business     655967037
         Luxury      1052569562
         Name: revenue_realized, dtype: int64
```

```
In [99]:  1  df_bookings.groupby('property_name')["revenue_realized"].sum().sort_values()

Out[99]: 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
```

INSIGHTS (AD HOC ANALYSIS)

9. Average rating per city

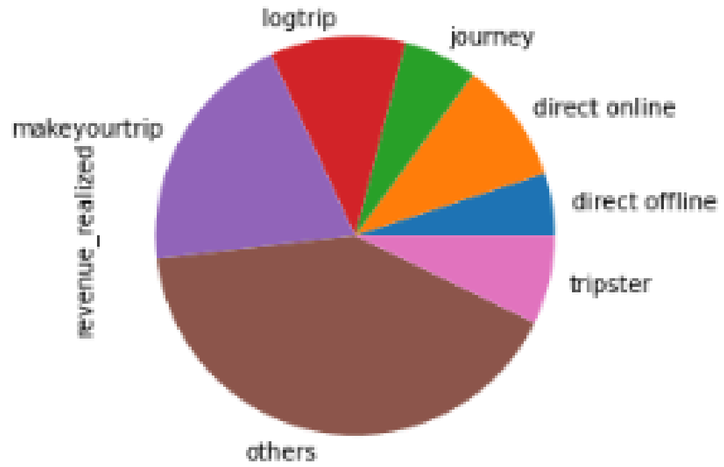
In [100]:
```
1 df_bookings.groupby("city")["ratings_given"].mean().round(2).sort_values()
```

Out[100]:
```
city
Bangalore    3.41
Mumbai       3.65
Hyderabad    3.66
Delhi        3.78
Name: ratings_given, dtype: float64
```

10. A pie chart of revenue realized per booking platform

In [101]:
```
1 df_bookings.groupby("booking_platform")["revenue_realized"].sum().plot(kind="pie")
```

Out[101]:
```
<AxesSubplot:ylabel='revenue_realized'>
```



Thank You