ATLIQ GRAND

# PYTHON HOSPITALITY PROJECT

# ATLIQ GRAND

- Insights presented by Devesh Singh

# CONTENT

- Problem Statement
- About the Dataset
- Data Cleaning
- Data Transformation
- Insights Generation

# Problem Statement

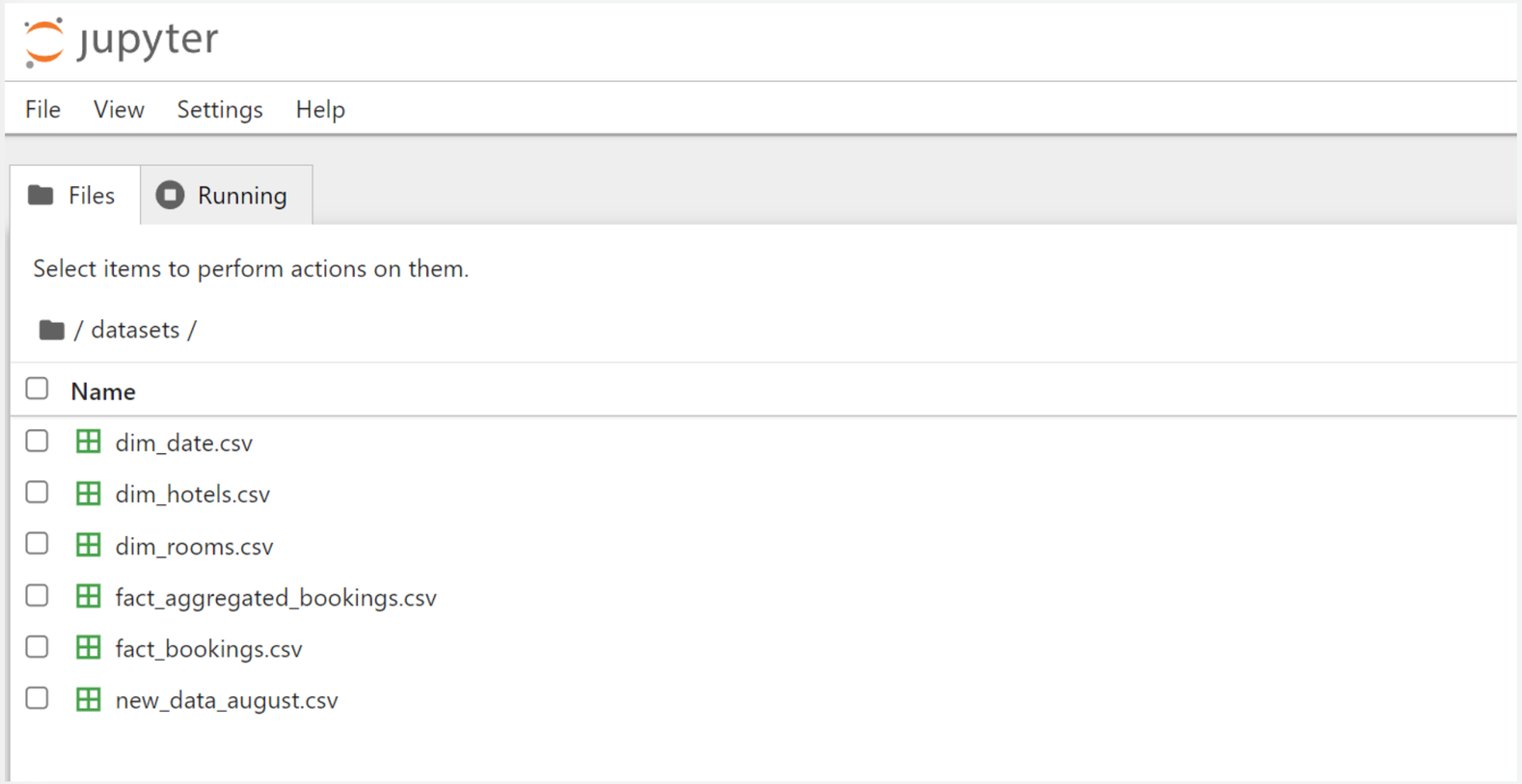
- AtliQ Grands owns multiple five-star hotels across India. They have been in the hospitality industry for the past 20 years. Due to strategic moves from other competitors and ineffective decision-making in management, AtliQ Grands are losing its market share and revenue in the luxury/business hotels category.

- As a strategic move, the managing director of AtliQ Grands wanted to regain their market share and revenue. Their revenue management team had decided to hire a 3rd party service provider to provide them with insights from their historical data.

- You are a data analyst who has been provided with sample data to provided the revenue insights to the team.

# About Dataset

- Team provided the 3 months booking data of AtliQ Grand having around 1.4 lakh records.
- Datset contains 3 dimension tables and to 2 fact tables.
- In between project, I was also provided with the ‘August’ month data to include it in the previous data.



## dim_date

```
# df_date = pd.read_csv('datasets/dim_date.csv')
df_date.head(3)
```

| | 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 |

## dim_hotels

```
# df_hotels = pd.read_csv('datasets/dim_hotels.csv')
df_hotels.head(3)
```

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

## dim_rooms

```
# df_rooms = pd.read_csv('datasets/dim_rooms.csv')
df_rooms.head(4)
```

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

# fact_bookings

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

```
[148]: Index(['booking_id', 'property_id', 'booking_date', 'check_in_date',
              'checkout_date', 'no_guests', 'room_category', 'booking_platform',
              'ratings_given', 'booking_status', 'revenue_generated',
              'revenue_realized'],
             dtype='object')
```

```
[163]: df_bookings
```

[163]:

| | 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 | 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 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **134585** | Jul312217564RT46 | 17564 | 29-07-22 | 31-07-22 | 3/8/2022 | 1.0 | RT4 | makeyourtrip | 2.0 | Checked Out | |
| **134586** | Jul312217564RT47 | 17564 | 30-07-22 | 31-07-22 | 1/8/2022 | -4.0 | RT4 | logtrip | 2.0 | Checked Out | |
| **134587** | Jul312217564RT48 | 17564 | 30-07-22 | 31-07-22 | 2/8/2022 | 1.0 | RT4 | tripster | NaN | Cancelled | |
| **134588** | Jul312217564RT49 | 17564 | 29-07-22 | 31-07-22 | 1/8/2022 | 2.0 | RT4 | logtrip | 2.0 | Checked Out | |
| **134589** | Jul312217564RT410 | 17564 | 31-07-22 | 31-07-22 | 1/8/2022 | 2.0 | RT4 | makeyourtrip | NaN | Cancelled | |

134590 rows × 12 columns

# fact_aggregated_bookings

```
# df_agg_bookings = pd.read_csv('datasets/fact_aggregated_bookings.csv')
df_agg_bookings
```

| | 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 |
| **...** | ... | ... | ... | ... | ... |
| **9195** | 16563 | 31-Jul-22 | RT4 | 13 | 18.0 |
| **9196** | 16559 | 31-Jul-22 | RT4 | 13 | 18.0 |
| **9197** | 17558 | 31-Jul-22 | RT4 | 3 | 6.0 |
| **9198** | 19563 | 31-Jul-22 | RT4 | 3 | 6.0 |
| **9199** | 17561 | 31-Jul-22 | RT4 | 3 | 4.0 |

# Data Cleaning

## ignored records containing -ve no_of_guests

**(1) Clean invalid guests**

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

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

```
# Here keeping the rows which has no_guest +ve

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

```
df_bookings.shape
```

```
(134578, 12)
```

## checked NaN (null values) in all columns

```
# checking/counting the null values for different columns of the df_bookings

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       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.
-- And It is also very much obvious from the fact that people do not consider to give a rating... -->
```

# replaced NaN with median

**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)**

```
df_agg_bookings.isnull().sum()
```

```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               2
dtype: int64
```

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

| | 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 |

```
# here I am replacing NaN values with the median. Btw, this decision depends on multiple factors...

df_agg_bookings.capacity.fillna(df_agg_bookings.capacity.median(), inplace=True)
```

```
df_agg_bookings.loc[[8,15]]
```

| | 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 |

# removed outliers by using std(standard deviation)

**(2) Outlier removal in revenue generated**

```
# checking min/max of revenue generated

df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

```
(6500, 28560000)
```

```
# checking mean and median

df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median()
```

```
(15378.036937686695, 13500.0)
```

```
# after seeing the max/min, mean and median of revenue_generated, I understood there are outlier values
# ...so now I will remove outlier values with the help of std(Standard Deviation)

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

```
higher_limit = avg + 3*std    # a mathematical way of finding the upper limit of an outlier
higher_limit
```

```
294498.50173207896
```

```
lower_limit = avg - 3*std    # ...similarly lower limit
lower_limit
```

```
-263742.4278567056
```

```
df_bookings[df_bookings.revenue_generated <= 0]    # no -ve revenue_generated value obtained
```

| booking_id | property_id | booking_date | check_in_date | checkout_date | no_guests | room_category | booking_platform | ratings_given | booking_status | revenue_gene |
|---|---|---|---|---|---|---|---|---|---|---|

```
# just checking the values which are greater than outlier values

df_bookings[df_bookings.revenue_generated > higher_limit]
```

```
# updating our dataframe, so that we have only those values which are lower than higher_limit

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

```
(134573, 12)
```

# Data Transformation

**Create occupancy percentage column**

```
df_agg_bookings.head(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 |

```
# creating occ_pct column by doing row-wise calculation...

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

created occupancy % column

converted it to percentage

Convert it to a percentage value

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

| | 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 |

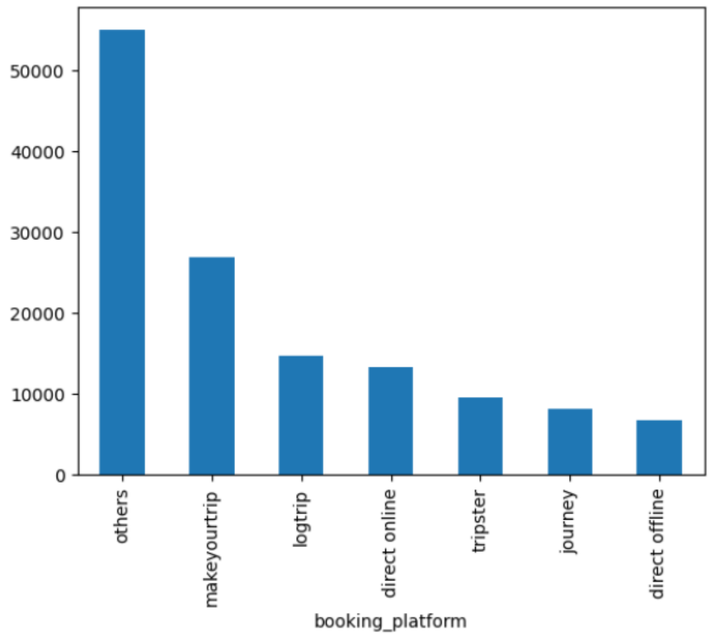
# Insights Generation

## Insight-1

**Insight1: Find out bookings by platform**

[18]:
```
df_bookings.booking_platform.value_counts().plot(kind="bar")
```

[18]:
```
<Axes: xlabel='booking_platform'>
```




## Insight-2

**Insight-2. Find out total bookings per property_id**

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

[35]:
```
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
```

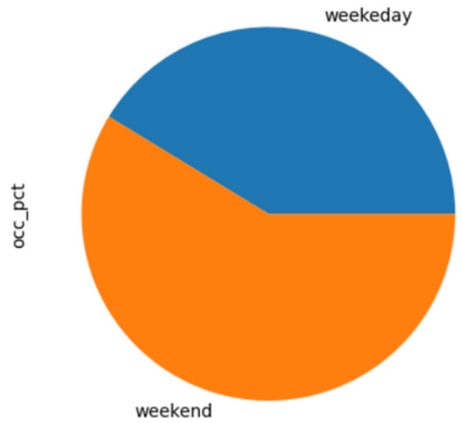
## Insight-3

**Insight-4. When was the occupancy better? Weekday or Weekend?**

```
# average occupancy rate on weekdays/weekends

df.groupby("day_type")["occ_pct"].mean().round(2).plot(kind='pie')
```

```
<Axes: ylabel='occ_pct'>
```




```
day_type
weekeday     50.88
weekend      72.34
Name: occ_pct, dtype: float64
```

## Insight-4

**Insight-3. What is an average occupancy rate in each of the room categories?**

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

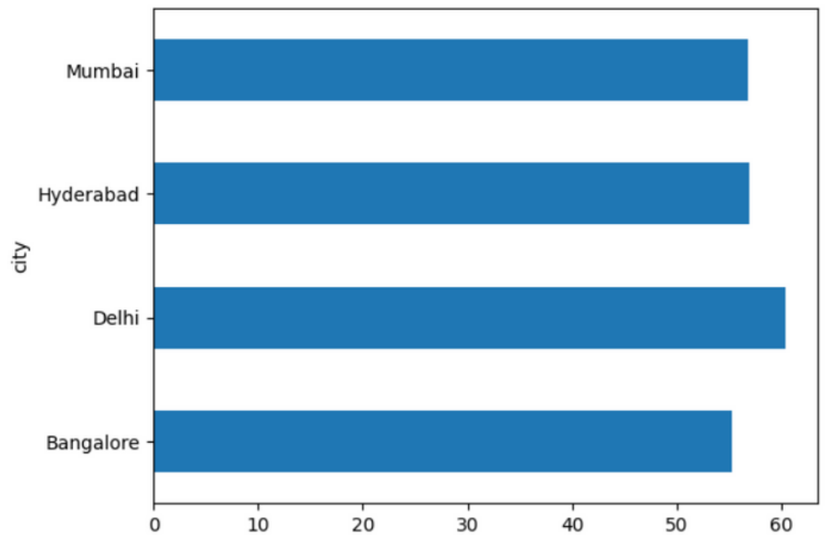
```
room_class
Elite           58.009756
Premium         58.028213
Presidential    59.277925
Standard        57.889643
Name: occ_pct, dtype: float64
```

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

**Insight-6. Print average occupancy rate per city**

```
df.groupby("city")["occ_pct"].mean().plot(kind='barh')
```

```
<Axes: ylabel='city'>
```




Insight-5

Insight-6

**Insight-5. In the month of June, what is the occupancy for different cities**

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

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

```
city
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Bangalore    56.44
Name: occ_pct, dtype: float64
```

Insight-7

**Insight-7: We got new data for the month of august. Append that to existing data**

[116]:
```
# # reading newly got data for august month

df_august = pd.read_csv("datasets/new_data_august.csv")
df_august.shape
```

[116]:
```
(7, 13)
```

[118]:
```
# adding August data into the dataframe by using 'concat' function

latest_df = pd.concat([df, df_august], ignore_index = True, axis = 0)
latest_df.tail(4)
```

[118]:

| | property_id | check_in_date | room_category | successful_bookings | capacity | occ_pct | room_class | property_name | category | city | date | mmm yy | week no | day_typ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **6500** | 19558 | 01-Aug-22 | RT1 | 30 | 40.0 | NaN | Standard | Atliq Grands | Luxury | Bangalore | NaN | Aug-22 | W 32 | weekeda |
| **6501** | 19560 | 01-Aug-22 | RT1 | 20 | 26.0 | NaN | Standard | Atliq City | Business | Bangalore | NaN | Aug-22 | W 32 | weekeda |
| **6502** | 17561 | 01-Aug-22 | RT1 | 18 | 26.0 | NaN | Standard | Atliq Blu | Luxury | Mumbai | NaN | Aug-22 | W 32 | weekeda |
| **6503** | 17564 | 01-Aug-22 | RT1 | 10 | 16.0 | NaN | Standard | Atliq Seasons | Business | Mumbai | NaN | Aug-22 | W 32 | weekeda |

Insight-8

**Insight-8. Print 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
```

Insight-9

**Insight-9. Print revenue realized per hotel type**

```
df_bookings_all.property_name.unique()
```

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

```
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
```

Insight-10

**Insight-10. Print average rating per city**

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

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

Insight-11

**Insight-11. Print a pie chart of revenue realized per booking platform**

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

```
booking_platform
direct offline     86374933
direct online     168948637
journey           102531334
logtrip           187494028
makeyourtrip      340814104
others            699306762
tripster          123066801
Name: revenue_realized, dtype: int64
```

# Thank You!