Jupyter Logo

==> 1. Data Import and Data Exploration

```
In [75]: import pandas as pd
```

```
In [76]: df_booking = pd.read_csv("dataset/fact_bookings.csv")
         df_booking.head(4)
```

Out[76]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_gues
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2

```
In [77]: # FIND TOTAL ROW & COLUMN COUNT
         df_booking.shape

Out[77]: (134590, 12)
```

FIND TYPE OF ROOM CATEGORIES

```
In [78]: df_booking.room_category.unique()

Out[78]: array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

FIND TYPE OF BOOKING PLATEFORMS AVAILABLE & THEIR C

```
 In [7]: df_booking.booking_platform.unique()

Out[7]: array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
               'journey', 'direct offline'], dtype=object)
```

FIND TOTAL BOOKING BY DIFFERENT PLATEFORMS

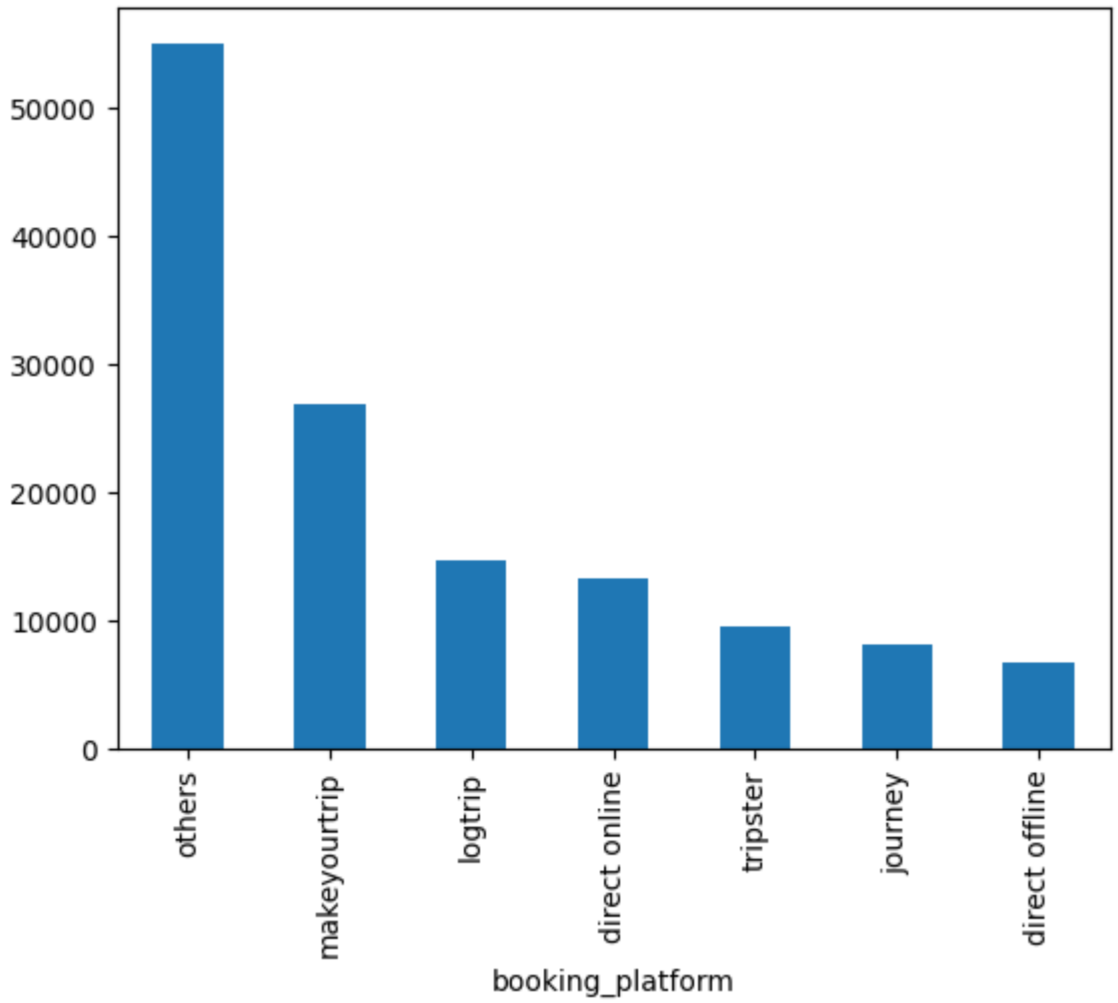
```
In [8]: df_booking.booking_platform.value_counts()
```

```
Out[8]: booking_platform
        others            55066
        makeyourtrip      26898
        logtrip           14756
        direct online     13379
        tripster           9630
        journey            8106
        direct offline     6755
        Name: count, dtype: int64
```

PROBLEM: CREATE A BAR CHART OF TOTAL BOOKING BY PLATFORM

```
In [10]: df_booking.booking_platform.value_counts().plot(kind = "bar")
```

```
Out[10]: <Axes: xlabel='booking_platform'>
```



```
In [79]: # GET OUT BASIC ANALYSIS OF TABLE
         df_booking.describe()
```

Out[79]:

	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

LOAD ALL THE OTHER FILES PRESENT IN PROJECT

In [80]:
```
df_date = pd.read_csv('dataset/dim_date.csv')
df_hotels = pd.read_csv('dataset/dim_hotels.csv')
df_rooms = pd.read_csv('dataset/dim_rooms.csv')
df_agg_bookings = pd.read_csv('dataset/fact_aggregated_bookings.csv')
```

In [13]:
```
# PRINT HOTEL TABLE FIRST 4 ROWS
df_hotels.head(4)
```

Out[13]:

	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

PROBLEM: FIND HOW MANY HOTEL ATLIQ HAS IN DIFFERENT CITIES, SORT BY DESC

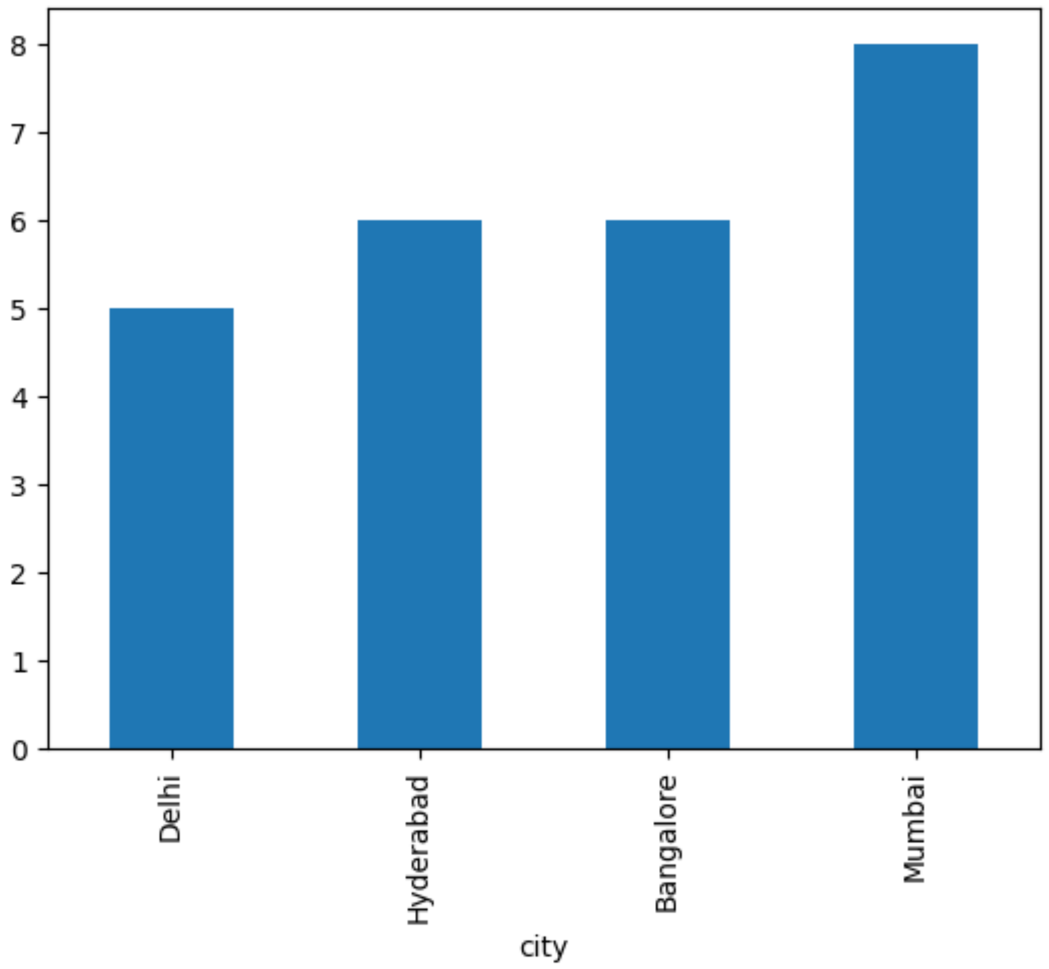
In [14]:
```
df_hotels.city.value_counts().sort_values()
```

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

PROBLEM: PLOT BAR CHART OF ATLIQ HOTELS DIFFERENT CITIES

In [15]:
```
# PLOT BAR CHART OF ABOVE DATA
df_hotels.city.value_counts().sort_values().plot(kind='bar') # CAN RESIZE
```

```
Out[15]: <Axes: xlabel='city'>
```



PROBLEM: Find out unique property ids in aggregate bookings dat

```
In [16]: df_booking.property_id.unique()
```

```
Out[16]: array([16558, 16559, 16560, 16561, 16562, 16563, 17558, 17559, 17560,
                17561, 17562, 17563, 18558, 18559, 18560, 18561, 18562, 18563,
                19558, 19559, 19560, 19561, 19562, 19563, 17564], dtype=int64)
```

==> 2. Data Cleaning

Clean invalid guests

```
In [17]: df_booking.describe()
```

Out[17]:

	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 [81]: df_booking[df_booking.no_guests<=0]
```

Out[81]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	n
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	
17924	May122218559RT44	18559	12/5/2022	12/5/2022	14-05-22	
18020	May122218561RT22	18561	8/5/2022	12/5/2022	14-05-22	
18119	May122218562RT311	18562	5/5/2022	12/5/2022	17-05-22	
18121	May122218562RT313	18562	10/5/2022	12/5/2022	17-05-22	
56715	Jun082218562RT12	18562	5/6/2022	8/6/2022	13-06-22	
119765	Jul202219560RT220	19560	19-07-22	20-07-22	22-07-22	
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	

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

```
In [82]: # Removing negative value and stoing in df_booking
         df_booking = df_booking[df_booking.no_guests>0]
         df_booking.shape

Out[82]: (134578, 12)
```

Outlier removal in revenue generated

```
In [84]: "MIN REVENUE: ", df_booking.revenue_generated.min(), "MAX REVENUE: ", df_l

Out[84]: ('MIN REVENUE: ', 6500, 'MAX REVENUE: ', 28560000)

In [85]: avg, std = df_booking.revenue_generated.mean() , df_booking.revenue_genera
         "Mean: ", avg,"STD: ", std

Out[85]: ('Mean: ', 15378.036937686695, 'STD: ', 93040.1549314641)
```

```
In [86]: higher_limit = avg + 3*std
         higher_limit

Out[86]: 294498.50173207896
```

FIND OUT VALUE CROSSING higher_limit

```
In [87]: df_booking[df_booking.revenue_generated > higher_limit]
```

Out[87]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	n
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	
111	May012216559RT32	16559	29-04-22	1/5/2022	2/5/2022	
315	May012216562RT22	16562	28-04-22	1/5/2022	4/5/2022	
562	May012217559RT118	17559	26-04-22	1/5/2022	2/5/2022	
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	

```
In [88]: # REMOVE ABOVE 5 OUTLIERS AND RESTORE THE DATA IN DF_BOOKING
         df_booking = df_booking[df_booking.revenue_generated < higher_limit]
         df_booking.shape
         # Earlier df_booking columns were (134578, 12) now -5

Out[88]: (134573, 12)

In [89]: df_booking.revenue_realized.describe()

Out[89]: 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 [91]: rev_realised_std_limit = df_booking.revenue_realized.mean() + 3*df_bookin
         "Revenue_Realized_Column_3rd_STD_limit: ",rev_realised_std_limit

Out[91]: ('Revenue_Realized_Column_3rd_STD_limit: ', 33479.358661845814)

In [92]: df_booking[df_booking.revenue_realized>rev_realised_std_limit]
```

Out[92]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	n
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	
139	May012216559RT43	16559	1/5/2022	1/5/2022	2/5/2022	
143	May012216559RT47	16559	28-04-22	1/5/2022	3/5/2022	
149	May012216559RT413	16559	24-04-22	1/5/2022	7/5/2022	
222	May012216560RT45	16560	30-04-22	1/5/2022	3/5/2022	
...	...	...	...	...	...	
134328	Jul312219560RT49	19560	31-07-22	31-07-22	2/8/2022	
134331	Jul312219560RT412	19560	31-07-22	31-07-22	1/8/2022	
134467	Jul312219562RT45	19562	28-07-22	31-07-22	1/8/2022	
134474	Jul312219562RT412	19562	25-07-22	31-07-22	6/8/2022	
134581	Jul312217564RT42	17564	31-07-22	31-07-22	1/8/2022	

1299 rows × 12 columns

One observation we can have in above dataframe is that all rooms are RT4 which means preside
suit. Now since RT4 is a luxurious room it is likely their rent will be higher. To make a fair analysis
need to do data analysis only on RT4 room types

In [93]:
```
df_rooms
```

Out[93]:

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

In [94]:
```
df_booking[df_booking.room_category=="RT4"].revenue_realized.describe()
```

Out[94]:
```
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
```

In [95]:
```
23439.308444 + 3*9048.599076
```

Out[95]:
```
50585.105672000005
```

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

PROBLEM: FIND THE NULL ROWS AND REMOVE AFTER VALID IF NEEDED.

In [96]:
```
df_booking.isnull().sum()
```

Out[96]:
```
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 r rows with null rating, we should not filter these values. Also we should not replace this rating with median or mean rating etc as rating can be left blank by customer

---

==> 3. Data Transformation

Create occupancy percentage column

In [97]:
```
df_agg_bookings.head()
```

Out[97]:

	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

CREATE A NEW COLUMN OF OCCUPANCY % I.E., successful_bookings / capacity

In [98]:
```
df_agg_bookings["Occupancy %"] = df_agg_bookings["successful_bookings"] /
df_agg_bookings.head(4)
```

Out[98]:

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupancy
0	16559	1-May-22	RT1	25	30.0	0.8333
1	19562	1-May-22	RT1	28	30.0	0.9333
2	19563	1-May-22	RT1	23	30.0	0.7666
3	17558	1-May-22	RT1	30	19.0	1.5789

In [99]:
```
# ROUNDING OCCUPANCY %
df_agg_bookings["Occupancy %"] = df_agg_bookings["Occupancy %"].apply(laml
df_agg_bookings
```

Out[99]:

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupa
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	[illegible]
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	

9200 rows × 6 columns

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

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

PROBLEM: FIND THE OCCUPANCY % BY ROOM_CATEGORY

In [100…
```
df_agg_bookings.groupby("room_category")["Occupancy %"].mean().round(2)
```

Out[100…
```
room_category
RT1    58.22
RT2    58.04
RT3    58.03
RT4    59.30
Name: Occupancy %, dtype: float64
```

In [101…
```
df_rooms
```

Out[101…

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

I don't understand RT1, RT2 etc. Print room categories such as Standard, Premium, Elite etc alo
average occupancy percentage

In [102…
```
df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_o
df.head()
```

Out[102…

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupancy %
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
3	17558	1-May-22	RT1	30	19.0	157.89
4	16558	1-May-22	RT1	18	19.0	94.74

In [103…
```
df.groupby("room_class")["Occupancy %"].mean().round(2)
```

Out[103…
```
room_class
Elite           58.04
Premium         58.03
Presidential    59.30
Standard        58.22
Name: Occupancy %, dtype: float64
```

In [104…
```
df.drop("room_id",axis=1,inplace=True)
df
```

Out[104…

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupa
0	16559	1-May-22	RT1	25	30.0	83
1	19562	1-May-22	RT1	28	30.0	93
2	19563	1-May-22	RT1	23	30.0	76
3	17558	1-May-22	RT1	30	19.0	157
4	16558	1-May-22	RT1	18	19.0	94
...	...	...	...	...	...	
9195	16563	31-Jul-22	RT4	13	18.0	72
9196	16559	31-Jul-22	RT4	13	18.0	72
9197	17558	31-Jul-22	RT4	3	6.0	50
9198	19563	31-Jul-22	RT4	3	6.0	50
9199	17561	31-Jul-22	RT4	3	4.0	75

9200 rows × 7 columns

In [105…
```
df_hotels.head(3)
```

Out[105…

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

In [106…
```
df1 = pd.merge(df,df_hotels, on="property_id")
df1.head(3)
```

Out[106…

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupancy %
0	16559	1-May-22	RT1	25	30.0	83.33
1	16559	2-May-22	RT1	20	30.0	66.67
2	16559	3-May-22	RT1	17	30.0	56.67

==> 4. Insights Generation

PROBLEM: Print average occupancy rate per city

In [107…
```
df1.groupby("city")["Occupancy %"].mean().round(2)
```

Out[107…
```
city
Bangalore    56.59
Delhi        61.61
Hyderabad    58.14
Mumbai       57.94
Name: Occupancy %, dtype: float64
```

PROBLEM: When was the occupancy better? Weekday or Weeker

In [108…
```
df1.head(3)
```

Out[108…

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupancy %
0	16559	1-May-22	RT1	25	30.0	83.33
1	16559	2-May-22	RT1	20	30.0	66.67
2	16559	3-May-22	RT1	17	30.0	56.67

In [109…
```
df_date.head(3)
```

Out[109…

	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

In [110…
```
df2 = pd.merge(df1, df_date, left_on="check_in_date", right_on="date")
df2.head(4)
```

Out[110…

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupancy %
0	16559	10-May-22	RT1	18	30.0	60.00
1	16559	10-May-22	RT2	25	41.0	60.98
2	16559	10-May-22	RT3	20	32.0	62.50
3	16559	10-May-22	RT4	13	18.0	72.22

In [111…
```
df2.groupby('day_type')["Occupancy %"].mean().round(2)
```

Out[111…
```
day_type
weekeday    50.90
weekend     72.39
Name: Occupancy %, dtype: float64
```

PROBLEM: In the month of June, what is the occupancy for differe cities

In [112…
```
df2["mmm yy"].unique()
```
Out[112…
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```
In [113…
```
df_june_22 = df2[df2["mmm yy"] == 'Jun 22']
df_june_22.head(3)
```
Out[113…

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupa
2200	16559	10-Jun-22	RT1	20	30.0	66
2201	16559	10-Jun-22	RT2	26	41.0	63
2202	16559	10-Jun-22	RT3	20	32.0	62

In [114…
```
df_june_22.groupby('city')['Occupancy %'].mean().round(2).sort_values(asc
```
Out[114…
```
city
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Bangalore    56.58
Name: Occupancy %, dtype: float64
```

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

In [115…
```
df_add_august = pd.read_csv("Dataset/new_data_august.csv")
df_add_august.head(4)
```
Out[115…

	property_id	property_name	category	city	room_category	room_class	check_i
0	16559	Atliq Exotica	Luxury	Mumbai	RT1	Standard	01-
1	19562	Atliq Bay	Luxury	Bangalore	RT1	Standard	01-
2	19563	Atliq Palace	Business	Bangalore	RT1	Standard	01-
3	19558	Atliq Grands	Luxury	Bangalore	RT1	Standard	01-

In [116…
```
df2.columns
```
Out[116…
```
Index(['property_id', 'check_in_date', 'room_category', 'successful_bookings
       'capacity', 'Occupancy %', 'room_class', 'property_name', 'category',
       'city', 'date', 'mmm yy', 'week no', 'day_type'],
      dtype='object')
```
In [117…
```
df_add_august.columns
```
Out[117…
```
Index(['property_id', 'property_name', 'category', 'city', 'room_category',
       'room_class', 'check_in_date', 'mmm yy', 'week no', 'day_type',
       'successful_bookings', 'capacity', 'Occupancy %'],
      dtype='object')
```

```
In [118… df2.shape
```

```
Out[118… (6500, 14)
```

```
In [119… df_add_august.shape
```

```
Out[119… (7, 13)
```

```
In [120… latest_df = pd.concat([df2,df_add_august], ignore_index=True,axis = 0)
         latest_df.tail(10)
```

Out[120…

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupa
6497	18560	31-Jul-22	RT2	34	40.0	85
6498	18560	31-Jul-22	RT3	17	24.0	70
6499	18560	31-Jul-22	RT4	12	15.0	80
6500	16559	01-Aug-22	RT1	30	30.0	100
6501	19562	01-Aug-22	RT1	21	30.0	70
6502	19563	01-Aug-22	RT1	23	30.0	76
6503	19558	01-Aug-22	RT1	30	40.0	75
6504	19560	01-Aug-22	RT1	20	26.0	76
6505	17561	01-Aug-22	RT1	18	26.0	69
6506	17564	01-Aug-22	RT1	10	16.0	62

```
In [121… latest_df.shape
```

```
Out[121… (6507, 14)
```

PROBLEM: CALCULATE THE REVENUE GENERATED PER CIT

```
In [122… df_booking.head(4)
```

Out[122…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_gues
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2
6	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2

```
In [123… df_hotels.head(4)
```

Out[123…

	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

PROBLEM. Print revenue realized per city

In [124…
```
df_revenue_city = pd.merge(df_booking, df_hotels, on = "property_id")
df_revenue_city.head(4)
```

Out[124…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_gues
0	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2
1	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4
2	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2
3	May012216558RT17	16558	28-04-22	1/5/2022	6/5/2022	2

In [125…
```
df_revenue_city.groupby('city')['revenue_realized'].sum()
```

Out[125…
```
city
Bangalore    420383550
Delhi        294404488
Hyderabad    325179310
Mumbai       668569251
Name: revenue_realized, dtype: int64
```

PROBLEM: MONTH BY MONTH REVENUE

In [126…
```
df_date["mmm yy"].unique()
```

Out[126…
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

In [127…
```
df_date.head(3)
```

Out[127…

	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

In [128…
```
latest_df.head(4)
```

Out[128…

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupancy %
0	16559	10-May-22	RT1	18	30.0	60.00
1	16559	10-May-22	RT2	25	41.0	60.98
2	16559	10-May-22	RT3	20	32.0	62.50
3	16559	10-May-22	RT4	13	18.0	72.22

In [129…

```
df_month_based_revenue = pd.merge(latest_df, df_date, left_on='check_in_da
df_month_based_revenue.head(4)
```

Out[129…

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupancy %
0	16559	10-May-22	RT1	18	30.0	60.00
1	16559	10-May-22	RT2	25	41.0	60.98
2	16559	10-May-22	RT3	20	32.0	62.50
3	16559	10-May-22	RT4	13	18.0	72.22

In [130…

```
df_booking.tail(4)
```

Out[130…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	
134587	Jul312217564RT48	17564	30-07-22	31-07-22	2/8/2022	
134588	Jul312217564RT49	17564	29-07-22	31-07-22	1/8/2022	
134589	Jul312217564RT410	17564	31-07-22	31-07-22	1/8/2022	

In [131…

```
latest_df.tail(3)
```

Out[131…

	property_id	check_in_date	room_category	successful_bookings	capacity	Occupa
6504	19560	01-Aug-22	RT1	20	26.0	76
6505	17561	01-Aug-22	RT1	18	26.0	69
6506	17564	01-Aug-22	RT1	10	16.0	62

In [132…

```
df_booking.head(3)
```

Out[132…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_gues
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	2
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	4
5	May012216558RT16	16558	1/5/2022	1/5/2022	3/5/2022	2

In [140…
```
df_date["mmm yy"].unique()
```

Out[140…
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

In [141…
```
df_booking.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 134573 entries, 1 to 134589
Data columns (total 12 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   booking_id         134573 non-null  object 
 1   property_id        134573 non-null  int64  
 2   booking_date       134573 non-null  object 
 3   check_in_date      134573 non-null  object 
 4   checkout_date      134573 non-null  object 
 5   no_guests          134573 non-null  float64
 6   room_category      134573 non-null  object 
 7   booking_platform   134573 non-null  object 
 8   ratings_given      56676 non-null   float64
 9   booking_status     134573 non-null  object 
 10  revenue_generated  134573 non-null  int64  
 11  revenue_realized   134573 non-null  int64  
dtypes: float64(2), int64(3), object(7)
memory usage: 13.3+ MB
```

In [142…
```
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     datetime64[ns]
 1   mmm yy    92 non-null     object        
 2   week no   92 non-null     object        
 3   day_type  92 non-null     object        
dtypes: datetime64[ns](1), object(3)
memory usage: 3.0+ KB
```

In [150…
```
df_date["date"] = pd.to_datetime(df_date["date"], errors='coerce')
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     datetime64[ns]
 1   mmm yy    92 non-null     object
 2   week no   92 non-null     object
 3   day_type  92 non-null     object
dtypes: datetime64[ns](1), object(3)
memory usage: 3.0+ KB
```

In [151…
```
df_booking["check_in_date"] = pd.to_datetime(df_booking["check_in_date"],

df_booking.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13795 entries, 0 to 13794
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   booking_id         13795 non-null  object
 1   property_id        13795 non-null  int64
 2   booking_date       13795 non-null  object
 3   check_in_date      13795 non-null  datetime64[ns]
 4   checkout_date      13795 non-null  object
 5   no_guests          13795 non-null  float64
 6   room_category      13795 non-null  object
 7   booking_platform   13795 non-null  object
 8   ratings_given      5673 non-null   float64
 9   booking_status     13795 non-null  object
 10  revenue_generated  13795 non-null  int64
 11  revenue_realized   13795 non-null  int64
 12  date_x             13795 non-null  datetime64[ns]
 13  mmm yy_x           13795 non-null  object
 14  week no_x          13795 non-null  object
 15  day_type_x         13795 non-null  object
 16  date_y             13795 non-null  datetime64[ns]
 17  mmm yy_y           13795 non-null  object
 18  week no_y          13795 non-null  object
 19  day_type_y         13795 non-null  object
dtypes: datetime64[ns](3), float64(2), int64(3), object(12)
memory usage: 2.1+ MB
```

In [152…
```
df_booking = pd.merge(df_booking,df_date, left_on='check_in_date',right_o
df_booking.head(3)
```

Out[152…

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_gues
0	May052216558RT11	16558	15-04-22	2022-05-05	7/5/2022	3
1	May052216558RT12	16558	30-04-22	2022-05-05	7/5/2022	2
2	May052216558RT13	16558	1/5/2022	2022-05-05	6/5/2022	3

3 rows × 24 columns

```
In [153… df_booking.groupby('mmm yy')['revenue_realized'].sum()

Out[153…  mmm yy
          Jul 22    60278496
          Jun 22    52903014
          May 22    60961428
          Name: revenue_realized, dtype: int64

 In [ ]:
```