

# hospitallity\_analytics

February 24, 2024

```
[112]: import pandas as pd
import matplotlib.pyplot as plt
```

---

## 0.0.1 0- Dataset

---

There are 5 csv file

- dim\_date.csv
  - dim\_hotels.csv
  - dim\_rooms.csv
  - fact\_aggregated\_bookings
  - fact\_bookings.csv.csv
- 

## 0.0.2 1- Data Exploration

---

```
[3]: df_bookings = pd.read_csv("datasets/fact_bookings.csv")
df_bookings.head()
```

```
[3]:
```

	booking_id	property_id	booking_date	check_in_date	checkout_date	\
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	
1	May012216558RT12	16558	30-04-22	1/5/2022	2/5/2022	
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	
4	May012216558RT15	16558	27-04-22	1/5/2022	2/5/2022	

	no_guests	room_category	booking_platform	ratings_given	booking_status	\
0	-3.0	RT1	direct online	1.0	Checked Out	
1	2.0	RT1	others	NaN	Cancelled	
2	2.0	RT1	logtrip	5.0	Checked Out	
3	-2.0	RT1	others	NaN	Cancelled	
4	4.0	RT1	direct online	5.0	Checked Out	

	revenue_generated	revenue_realized
0	10010	10010
1	9100	3640
2	9100000	9100
3	9100	3640
4	10920	10920

```
[4]: df_bookings.shape
```

```
[4]: (134590, 12)
```

```
[5]: df_bookings.room_category.unique()
```

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

```
[6]: df_bookings.booking_platform.unique()
```

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

```
[7]: df_bookings[df_bookings['booking_platform']=='logtrip'].shape
```

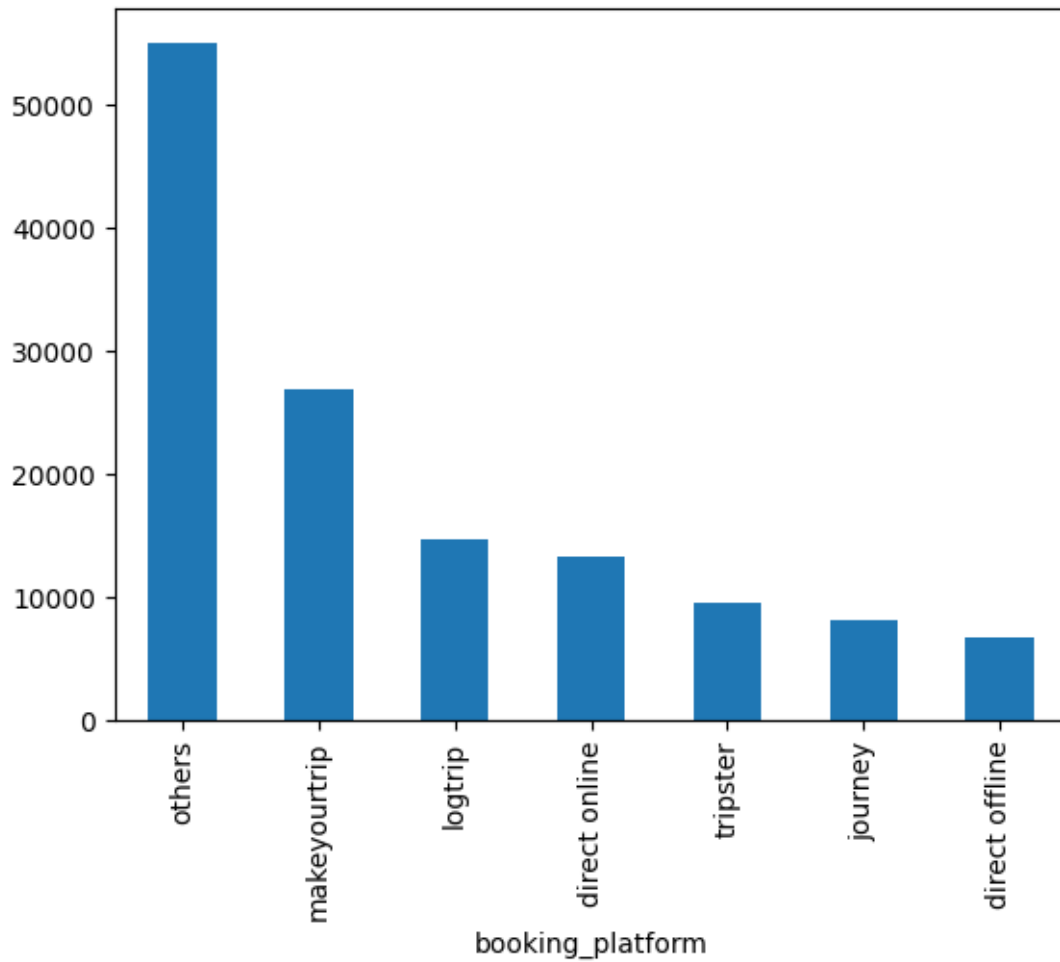
```
[7]: (14756, 12)
```

```
[8]: df_bookings.booking_platform.value_counts()
```

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

```
[9]: df_bookings.booking_platform.value_counts().plot(kind='bar')
```

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



```
[10]: df_bookings.describe()
```

```
[10]:
```

	property_id	no_guests	ratings_given	revenue_generated \
count	134590.000000	134587.000000	56683.000000	1.345900e+05
mean	18061.113493	2.036170	3.619004	1.537805e+04
std	1093.055847	1.034885	1.235009	9.303604e+04
min	16558.000000	-17.000000	1.000000	6.500000e+03
25%	17558.000000	1.000000	3.000000	9.900000e+03
50%	17564.000000	2.000000	4.000000	1.350000e+04
75%	18563.000000	2.000000	5.000000	1.800000e+04
max	19563.000000	6.000000	5.000000	2.856000e+07

	revenue_realized
count	134590.000000
mean	12696.123256
std	6928.108124
min	2600.000000

```
25%          7600.000000
50%         11700.000000
75%         15300.000000
max          45220.000000
```

```
[11]: df_bookings.booking_platform.describe()
```

```
[11]: count      134590
      unique         7
      top      others
      freq      55066
      Name: booking_platform, dtype: object
```

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

```
[12]: (6500, 28560000)
```

```
[13]: df_date = pd.read_csv('datasets/dim_date.csv')
      df_hotels = pd.read_csv('datasets/dim_hotels.csv')
      df_room = pd.read_csv('datasets/dim_rooms.csv')
      df_agg_bookings = pd.read_csv('datasets/fact_aggregated_bookings.csv')
```

```
[14]: df_hotels.head(3)
```

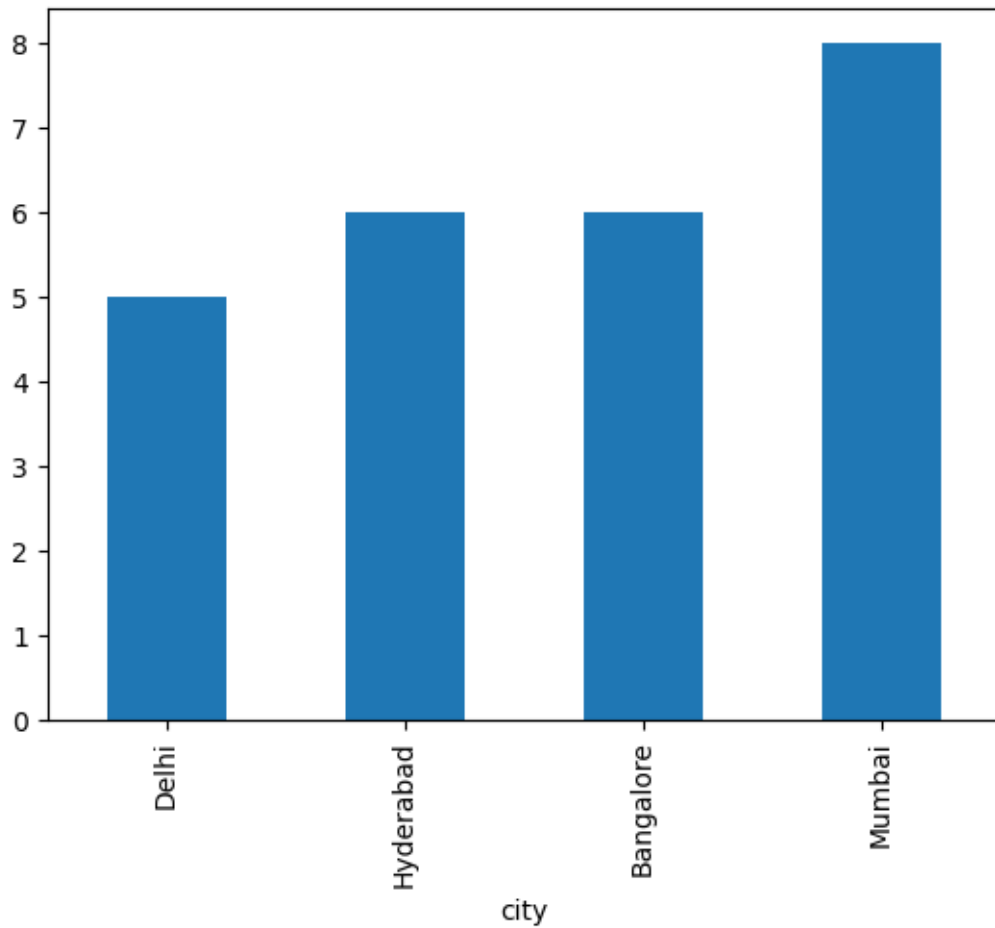
```
[14]:   property_id  property_name  category  city
0         16558    Atliq Grands   Luxury  Delhi
1         16559    Atliq Exotica   Luxury  Mumbai
2         16560     Atliq City  Business  Delhi
```

```
[15]: df_hotels.category.value_counts()
```

```
[15]: category
      Luxury      16
      Business     9
      Name: count, dtype: int64
```

```
[16]: df_hotels.city.value_counts().sort_values().plot(kind='bar')
```

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



```
[ ]:
```

### 0.0.3 2- Data Cleaning

#### (1) Clean invalid guests

```
[17]: df_bookings.describe()
```

```
[17]:
```

	property_id	no_guests	ratings_given	revenue_generated \
count	134590.000000	134587.000000	56683.000000	1.345900e+05
mean	18061.113493	2.036170	3.619004	1.537805e+04
std	1093.055847	1.034885	1.235009	9.303604e+04
min	16558.000000	-17.000000	1.000000	6.500000e+03
25%	17558.000000	1.000000	3.000000	9.900000e+03
50%	17564.000000	2.000000	4.000000	1.350000e+04

75%	18563.000000	2.000000	5.000000	1.800000e+04
max	19563.000000	6.000000	5.000000	2.856000e+07

	revenue_realized
count	134590.000000
mean	12696.123256
std	6928.108124
min	2600.000000
25%	7600.000000
50%	11700.000000
75%	15300.000000
max	45220.000000

```
[18]: # we notice that min value of no_guests is negative (-17)
```

```
[19]: df_bookings[df_bookings['no_guests']<=0]
```

```
[19]:
```

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

	checkout_date	no_guests	room_category	booking_platform	ratings_given	\
0	2/5/2022	-3.0	RT1	direct online	1.0	
3	2/5/2022	-2.0	RT1	others	NaN	
17924	14-05-22	-10.0	RT4	direct online	NaN	
18020	14-05-22	-12.0	RT2	makeyourtrip	NaN	
18119	17-05-22	-6.0	RT3	direct offline	5.0	
18121	17-05-22	-4.0	RT3	direct online	NaN	
56715	13-06-22	-17.0	RT1	others	NaN	
119765	22-07-22	-1.0	RT2	others	NaN	
134586	1/8/2022	-4.0	RT4	logtrip	2.0	

	booking_status	revenue_generated	revenue_realized
0	Checked Out	10010	10010
3	Cancelled	9100	3640
17924	No Show	20900	20900
18020	Cancelled	9000	3600
18119	Checked Out	16800	16800
18121	Cancelled	14400	5760
56715	Checked Out	6500	6500

119765	Checked Out	13500	13500
134586	Checked Out	38760	38760

```
[20]: # remove that datapoints as they are not huge ( just 12)
df_bookings = df_bookings[df_bookings['no_guests'] > 0]
df_bookings.shape
```

```
[20]: (134578, 12)
```

## (2) Outlier removal in revenue generated

```
[21]: # looking at revenue_generated column and notice some issues
      # One can not paid 28560000 for a single booking.
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

```
[21]: (6500, 28560000)
```

```
[22]: # to tackle this issue, we gonna remove outlier ( datapoints that are greater
      ↪ than 3*std)

      #--> Getting avg, std of revenue_generated

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

```
[22]: (15378.036937686695, 93040.1549314641)
```

```
[23]: higher_limit = avg + 3*std # q_3
higher_limit
```

```
[23]: 294498.50173207896
```

```
[24]: lower_limit = avg - 3*std #
lower_limit
```

```
[24]: -263742.4278567056
```

```
[25]: # Now, look at outliers : datapoints where revenue_generated > higher_limit
df_bookings[df_bookings['revenue_generated'] > higher_limit]
```

```
[25]:
```

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

	checkout_date	no_guests	room_category	booking_platform	ratings_given	\
2	4/5/2022	2.0	RT1	logtrip	5.0	
111	2/5/2022	6.0	RT3	direct online	NaN	
315	4/5/2022	2.0	RT2	direct offline	3.0	
562	2/5/2022	2.0	RT1	others	NaN	
129176	29-07-22	2.0	RT2	direct online	3.0	

	booking_status	revenue_generated	revenue_realized
2	Checked Out	9100000	9100
111	Checked Out	28560000	28560
315	Checked Out	12600000	12600
562	Cancelled	2000000	4420
129176	Checked Out	10000000	12600

```
[26]: # Remove outliers from dataset
# As revenue generated is positive for all datapoints in our dataset, we don't
      ↪ care about lower_limit

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

```
[26]: (134573, 12)
```

```
[ ]:
```

```
[27]: ## Same analysis for revenue_realized column
df_bookings.revenue_realized.describe()
```

```
[27]: 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
```

```
[28]: higher_limit_rev_realized = df_bookings['revenue_realized'].mean() +
      ↪ 3*df_bookings['revenue_realized'].std()
higher_limit_rev_realized
```

```
[28]: 33479.358661845814
```

```
[29]: # Now, let's find out datapoints that are far from 'higher_limit_rev_realized'
### From the output, we can notice that there are many rows, and most of them
      ↪ has 'RT4' as room_category
```



```
#### Thus, we can't consider this datapoints as outliers
#### Instead, let look at statics about 'RT4' room category
```

```
df_bookings[df_bookings['revenue_realized'] > higher_limit_rev_realized]
```

```
[29]:
```

	booking_id	property_id	booking_date	check_in_date	\
137	May012216559RT41	16559	27-04-22	1/5/2022	
139	May012216559RT43	16559	1/5/2022	1/5/2022	
143	May012216559RT47	16559	28-04-22	1/5/2022	
149	May012216559RT413	16559	24-04-22	1/5/2022	
222	May012216560RT45	16560	30-04-22	1/5/2022	
...	...	...	...	...	
134328	Jul1312219560RT49	19560	31-07-22	31-07-22	
134331	Jul1312219560RT412	19560	31-07-22	31-07-22	
134467	Jul1312219562RT45	19562	28-07-22	31-07-22	
134474	Jul1312219562RT412	19562	25-07-22	31-07-22	
134581	Jul1312217564RT42	17564	31-07-22	31-07-22	

	checkout_date	no_guests	room_category	booking_platform	ratings_given	\
137	7/5/2022	4.0	RT4	others	NaN	
139	2/5/2022	6.0	RT4	tripster	3.0	
143	3/5/2022	3.0	RT4	others	5.0	
149	7/5/2022	5.0	RT4	logtrip	NaN	
222	3/5/2022	5.0	RT4	others	3.0	
...	...	...	...	...	...	
134328	2/8/2022	6.0	RT4	direct online	5.0	
134331	1/8/2022	6.0	RT4	others	2.0	
134467	1/8/2022	6.0	RT4	makeyourtrip	4.0	
134474	6/8/2022	5.0	RT4	direct offline	5.0	
134581	1/8/2022	4.0	RT4	makeyourtrip	4.0	

	booking_status	revenue_generated	revenue_realized
137	Checked Out	38760	38760
139	Checked Out	45220	45220
143	Checked Out	35530	35530
149	Checked Out	41990	41990
222	Checked Out	34580	34580
...	...	...	...
134328	Checked Out	39900	39900
134331	Checked Out	39900	39900
134467	Checked Out	39900	39900
134474	Checked Out	37050	37050
134581	Checked Out	38760	38760

[1299 rows x 12 columns]

```
[30]:
```

```
[31]: ##### let look at statics about 'RT4' room category
# First remark, 'RT4' room are 'Presidential' class
df_room
```

```
[31]:   room_id   room_class
0     RT1     Standard
1     RT2        Elite
2     RT3        Premium
3     RT4  Presidential
```

```
[32]: ### Describe revenue_realized statistic for 'RT4' (presidential suit) from
      ↪df_bookings'
```

```
df_bookings[df_bookings.room_category == 'RT4'].revenue_realized.describe()
```

```
[32]: 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
```

```
[33]: # Remarks:
      # mean = 23439
      # max = 45220
      # std = 9048
      # higher_limit = mean + 3*std => higher_limit =
higher_limit_RT4 = 23439 + 3*9048
higher_limit_RT4
```

```
[33]: 50583
```

```
[34]: # higher_limit_RT4 = 50583 means that outliers are datapoints where
      # revenue_realized > 50583
      # but, max = 45220; so there is no outlier for 'revenue_realized'
```

## 2-1 : Handling 'NaN' Values

```
[35]: # It is ok to have 'NaN' values for 'ratings_given' as not every customers give
      ↪rating after in the hotel
      # So, we don't remove them.
df_bookings.isnull().sum()
```

```
[35]: 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

```

```
[ ]:
```

### 0.0.4 3- Data Transformation

Create occupancy percentage column

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

```

[36]:   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

```

```

[37]: df_agg_bookings["occ_pct"] = df_agg_bookings['successful_bookings']/
      ↪df_agg_bookings['capacity']
df_agg_bookings.head(3)

```

```

[37]:   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

      occ_pct
0  0.833333
1  0.933333
2  0.766667

```

Convert it to a percentage value

```

[38]: # convert value to %
df_agg_bookings['occ_pct'] = df_agg_bookings.occ_pct.apply(lambda x:
      ↪round(x*100,2))
df_agg_bookings.head(3)

```

```
[38]: 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

      occ_pct
0      83.33
1      93.33
2      76.67
```

```
[ ]:
```

#### 0.0.5 4- Insights Generation

```
[111]: # Utility function: convert amount to millio
def amount_to_million(amount):
    million_value = round(amount/10**6,2)
    return million_value
```

a. What is an average occupancy rate in each of the room categories ?

```
[39]: df_agg_bookings.groupby('room_category').occ_pct.mean()
```

```
[39]: room_category
RT1      58.224247
RT2      58.040278
RT3      58.028213
RT4      59.300461
Name: occ_pct, dtype: float64
```

```
[40]: # let's do the something and print Room names instead of their categories

df = pd.merge(df_agg_bookings, df_room, left_on='room_category',
              ↪right_on='room_id')
df.drop('room_id', axis=1, inplace=True)
df.head(3)
```

```
[40]: 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

      occ_pct room_class
0      83.33   Standard
1      93.33   Standard
```

```
2    76.67    Standard
```

```
[41]: df.groupby('room_class').occ_pct.mean().round(2)
```

```
[41]: room_class
Elite          58.04
Premium        58.03
Presidential   59.30
Standard       58.22
Name: occ_pct, dtype: float64
```

#### b. Print average occupancy rate per city

```
[42]: #city info isn't in df, so let's make join with hotel_df
df = pd.merge(df, df_hotels, on='property_id', how='left')
df.head(3)
```

```
[42]:   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

   occ_pct room_class property_name category      city
0    83.33  Standard  Atliq Exotica   Luxury  Mumbai
1    93.33  Standard      Atliq Bay   Luxury  Bangalore
2    76.67  Standard  Atliq Palace  Business  Bangalore
```

```
[43]: df.city.isnull().sum()
```

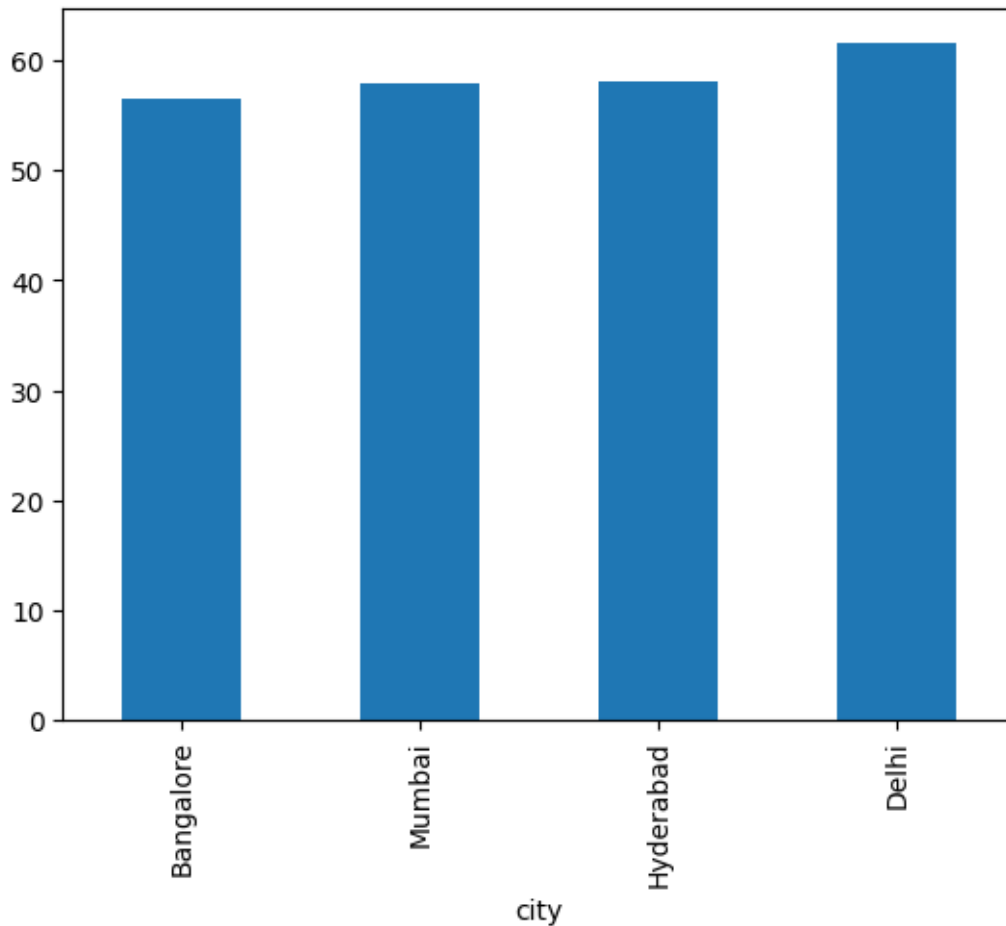
```
[43]: 0
```

```
[44]: df.groupby('city').occ_pct.mean().round(2)
```

```
[44]: city
Bangalore    56.59
Delhi        61.61
Hyderabad    58.14
Mumbai       57.94
Name: occ_pct, dtype: float64
```

```
[45]: df.groupby('city').occ_pct.mean().round(2).sort_values().plot(kind='bar')
```

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



c. When was the occupancy better? Weekday or Weekend?

```
[46]: df_date.head(3)
```

```
[46]:      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
```

```
[47]: #daytype info is in df_date, so let's make join between df & df_date

df = pd.merge(df, df_date, left_on='check_in_date', right_on='date')
#df.drop('date'
df.head(3)
```

```
[47]:   property_id  check_in_date  room_category  successful_bookings  capacity  \
0         18560      10-May-22             RT1                   19        30.0
1         19562      10-May-22             RT1                   18        30.0
```

2	19563	10-May-22	RT1	16	30.0
---	-------	-----------	-----	----	------

	occ_pct	room_class	property_name	category	city	date	mmm yy \
0	63.33	Standard	Atliq City	Business	Hyderabad	10-May-22	May 22
1	60.00	Standard	Atliq Bay	Luxury	Bangalore	10-May-22	May 22
2	53.33	Standard	Atliq Palace	Business	Bangalore	10-May-22	May 22

	week no	day_type
0	W 20	weekeday
1	W 20	weekeday
2	W 20	weekeday

```
[48]: df.groupby('day_type').occ_pct.max()
```

```
[48]: day_type
weekeday    145.83
weekend      128.21
Name: occ_pct, dtype: float64
```

**c: In the month of June, what is the occupancy for different cities**

```
[49]: df['mmm yy'].unique()
```

```
[49]: array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

```
[50]: df_june = df[df['mmm yy']=='Jun 22']
df_june.groupby('city').occ_pct.mean().round(2)
```

```
[50]: city
Bangalore    56.58
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Name: occ_pct, dtype: float64
```

**d: Add august datas**

```
[51]: df_august = pd.read_csv('datasets/new_data_august.csv')
df_august.shape
```

```
[51]: (7, 13)
```

```
[52]: df.shape
```

```
[52]: (6500, 14)
```

```
[53]: df.columns
```

```
[53]: Index(['property_id', 'check_in_date', 'room_category', 'successful_bookings',
        'capacity', 'occ_pct', 'room_class', 'property_name', 'category',
        'city', 'date', 'mmm yy', 'week no', 'day_type'],
        dtype='object')
```

```
[54]: df_august.columns
```

```
[54]: Index(['property_id', 'property_name', 'category', 'city', 'room_category',
        'room_class', 'check_in_date', 'mmm yy', 'week no', 'day_type',
        'successful_bookings', 'capacity', 'occ%'],
        dtype='object')
```

```
[55]: latest_df = pd.concat([df, df_august], ignore_index=True, axis=0)
      print(latest_df.shape)
      df.head(3)
```

```
(6507, 15)
```

```
[55]:   property_id  check_in_date  room_category  successful_bookings  capacity \
0         18560      10-May-22           RT1              19         30.0
1         19562      10-May-22           RT1              18         30.0
2         19563      10-May-22           RT1              16         30.0

      occ_pct  room_class  property_name  category  city  date  mmm yy \
0    63.33   Standard   Atliq City  Business  Hyderabad  10-May-22  May 22
1    60.00   Standard   Atliq Bay   Luxury  Bangalore  10-May-22  May 22
2    53.33   Standard   Atliq Palace Business  Bangalore  10-May-22  May 22

      week no  day_type
0      W 20  weekday
1      W 20  weekday
2      W 20  weekday
```

#### e- Print revenue realized per City

```
[56]: df_bookings.head(2)
```

```
[56]:   booking_id  property_id  booking_date  check_in_date  checkout_date \
1  May012216558RT12         16558      30-04-22      1/5/2022      2/5/2022
4  May012216558RT15         16558      27-04-22      1/5/2022      2/5/2022

      no_guests  room_category  booking_platform  ratings_given  booking_status \
1           2.0           RT1           others           NaN      Cancelled
4           4.0           RT1  direct online           5.0      Checked Out

      revenue_generated  revenue_realized
1              9100              3640
4             10920             10920
```



```
[57]: df_hotels.sample(2)
```

```
[57]:   property_id  property_name  category    city
13         18559  Atliq Exotica   Luxury  Hyderabad
2          16560    Atliq City  Business    Delhi
```

```
[58]: # As cities infos aren't in df_bookings, we need to merge with df_hotels

df_full_bookings = pd.merge(df_bookings, df_hotels, on='property_id',
                             how='left')
df_full_bookings.sample(3)
```

```
[58]:   booking_id  property_id  booking_date  check_in_date  checkout_date \
44359  May302218562RT44         18562    25-05-22    30-05-22    1/6/2022
88660  Jun302217561RT11         17561    27-06-22    30-06-22    1/7/2022
21142  May142218561RT26         18561    7/5/2022    14-05-22    15-05-22

   no_guests  room_category  booking_platform  ratings_given  booking_status \
44359        2.0           RT4           others           5.0    Checked Out
88660        2.0           RT1           others           NaN    Cancelled
21142        3.0           RT2           others           NaN    Checked Out

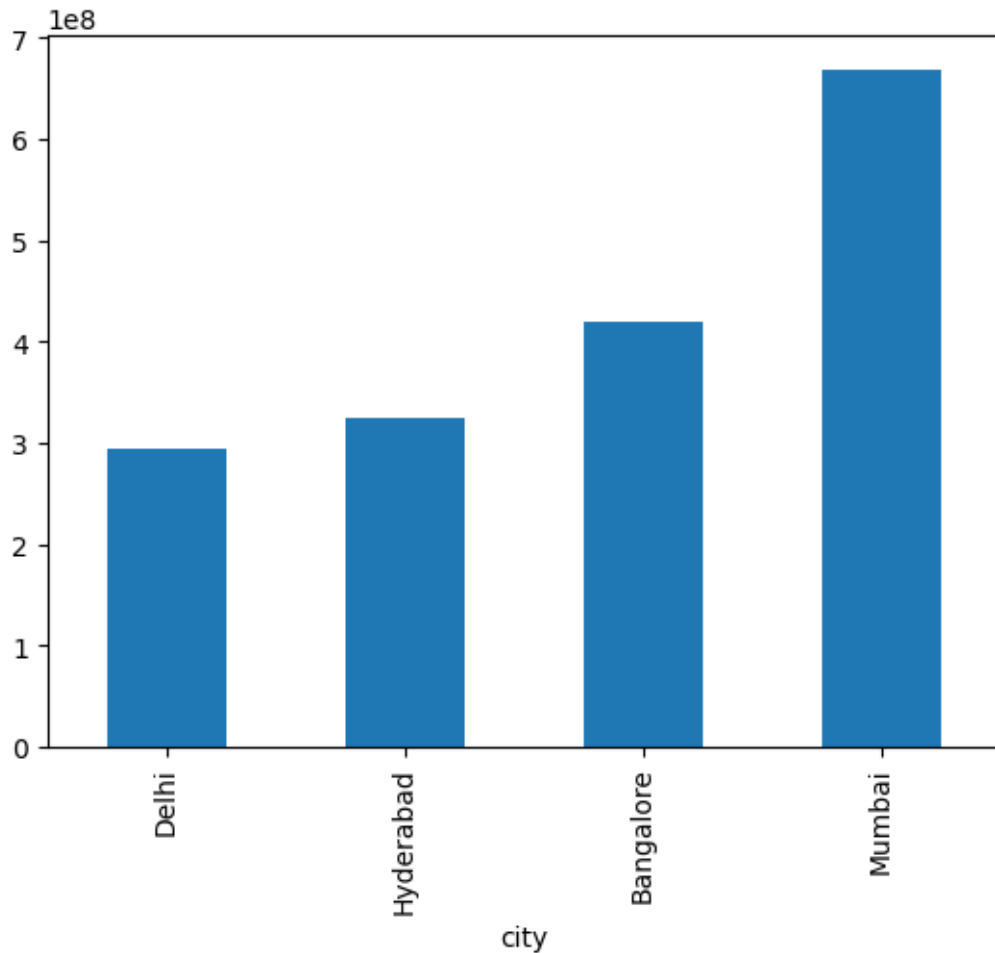
   revenue_generated  revenue_realized  property_name  category    city
44359             19000             19000    Atliq Bay   Luxury  Hyderabad
88660             11050              4420    Atliq Blu   Luxury    Mumbai
21142              9900              9900    Atliq Blu   Luxury  Hyderabad
```

```
[59]: df_full_bookings.groupby('city').revenue_realized.sum().sort_values()
```

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

```
[60]: df_full_bookings.groupby('city').revenue_realized.sum().sort_values().
      plot(kind='bar')
```

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



f- Print month by month revenue

```
[61]: df_full_bookings.sample(2)
```

```
[61]:      booking_id  property_id booking_date check_in_date \
125505  Jul242217564RT29      17564    24-07-22    24-07-22
64404   Jun132217559RT24      17559    24-05-22    13-06-22

      checkout_date  no_guests room_category booking_platform  ratings_given \
125505    27-07-22        1.0         RT2    direct online           NaN
64404    16-06-22        2.0         RT2         tripster           5.0

      booking_status  revenue_generated  revenue_realized  property_name \
125505    Cancelled             15300             6120  Atliq Seasons
64404    Checked Out             15300             15300  Atliq Exotica

      category  city
```

```
125505 Business Mumbai
64404    Luxury Mumbai
```

```
[62]: df_date['mmm yy'].unique()
```

```
[62]: array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

```
[63]: # date data are not in df_full_bookings,
pd.merge(df_full_bookings, df_date, left_on='check_in_date', right_on='date')
```

```
[63]: Empty DataFrame
Columns: [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, property_name, category, city, date, mmm
yy, week no, day_type]
Index: []
```

```
[64]: ## We remark that the output from the previous merge is empty,
      # this is because date and check_in_date are in different format
      # let's tackle this issue
df_full_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134573 entries, 0 to 134572
Data columns (total 15 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
12  property_name         134573 non-null object
13  category              134573 non-null object
14  city                 134573 non-null object
dtypes: float64(2), int64(3), object(10)
memory usage: 15.4+ MB
```

```
[65]: 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	object
1	mmm yy	92 non-null	object
2	week no	92 non-null	object
3	day_type	92 non-null	object

dtypes: object(4)  
memory usage: 3.0+ KB

```
[67]: # we need to convert 'check_in_date' and 'date' columns to 'datetime'
```

```
df_date['date'] = pd.to_datetime(df_date['date'], format='mixed')
df_date.head(2)
```

```
[67]:      date  mmm yy week no  day_type
0 2022-05-01  May 22   W 19  weekend
1 2022-05-02  May 22   W 19  weekday
```

```
[68]: df_full_bookings['check_in_date'] = pd.
      ↪to_datetime(df_full_bookings['check_in_date'], format='mixed')
df_full_bookings.sample(5)
```

```
[68]:      booking_id  property_id booking_date check_in_date \
45373  May312218560RT213      18560    29-05-22    2022-05-31
87732   Jun292218563RT19      18563    27-06-22    2022-06-29
106394  Jul112217561RT16      17561    9/7/2022    2022-11-07
44850   May312216561RT12      16561    28-05-22    2022-05-31
34880   May242217561RT220      17561    21-05-22    2022-05-24

      checkout_date  no_guests room_category booking_platform ratings_given \
45373    1/6/2022      2.0      RT2      others      NaN
87732    30-06-22      1.0      RT1  makeyourtrip      5.0
106394    16-07-22      4.0      RT1      tripster      NaN
44850    2/6/2022      2.0      RT1  direct online      3.0
34880    30-05-22      3.0      RT2      others      NaN

      booking_status  revenue_generated  revenue_realized property_name \
45373    Cancelled      9000      3600    Atliq City
87732    Checked Out      6500      6500    Atliq Palace
106394    Cancelled     13260      5304    Atliq Blu
44850    Checked Out      9100      9100    Atliq Blu
34880    Cancelled     16830      6732    Atliq Blu

      category      city
45373  Business  Hyderabad
87732  Business  Hyderabad
106394  Luxury   Mumbai
```

44850	Luxury	Delhi
34880	Luxury	Mumbai

```
[69]: df_full_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134573 entries, 0 to 134572
Data columns (total 15 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 datetime64[ns]
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
12  property_name         134573 non-null object
13  category              134573 non-null object
14  city                  134573 non-null object
dtypes: datetime64[ns](1), float64(2), int64(3), object(9)
memory usage: 15.4+ MB
```

```
[70]: df_full_bookings = pd.merge(df_full_bookings, df_date, left_on='check_in_date',
    ↪right_on='date')
```

```
[71]: df_full_bookings.sample(2)
```

```
[71]:
```

	booking_id	property_id	booking_date	check_in_date	checkout_date	\
4519	May072219561RT23	19561	7/5/2022	2022-07-05	9/5/2022	
48211	Jun202218560RT24	18560	17-06-22	2022-06-20	21-06-22	

	no_guests	room_category	booking_platform	ratings_given	booking_status	\
4519	4.0	RT2	others	NaN	Cancelled	
48211	4.0	RT2	others	NaN	No Show	

	revenue_generated	revenue_realized	property_name	category	city	\
4519	16200	6480	Atliq Blu	Luxury	Bangalore	
48211	10800	10800	Atliq City	Business	Hyderabad	

	date	mmm	yy	week no	day_type
4519	2022-07-05	Jul	22	W 28	weekeday
48211	2022-06-20	Jun	22	W 26	weekeday

```
[72]: # month by month revenue
df_full_bookings.groupby('mmm yy').revenue_realized.sum().sort_values()
```

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

```
[79]: #df_full_bookings.groupby('mmm yy').revenue_realized.sum().sort_values().
      ↪plot(kind='pie')
```

```
[78]: import mpld3

# Group the data and calculate the sum of revenue_realized
grouped_data = df_full_bookings.groupby('mmm yy').revenue_realized.sum().
      ↪sort_values()

# Create a pie chart with matplotlib
fig, ax = plt.subplots()
ax.pie(grouped_data.values, labels=grouped_data.index, autopct='%1.1f%%')

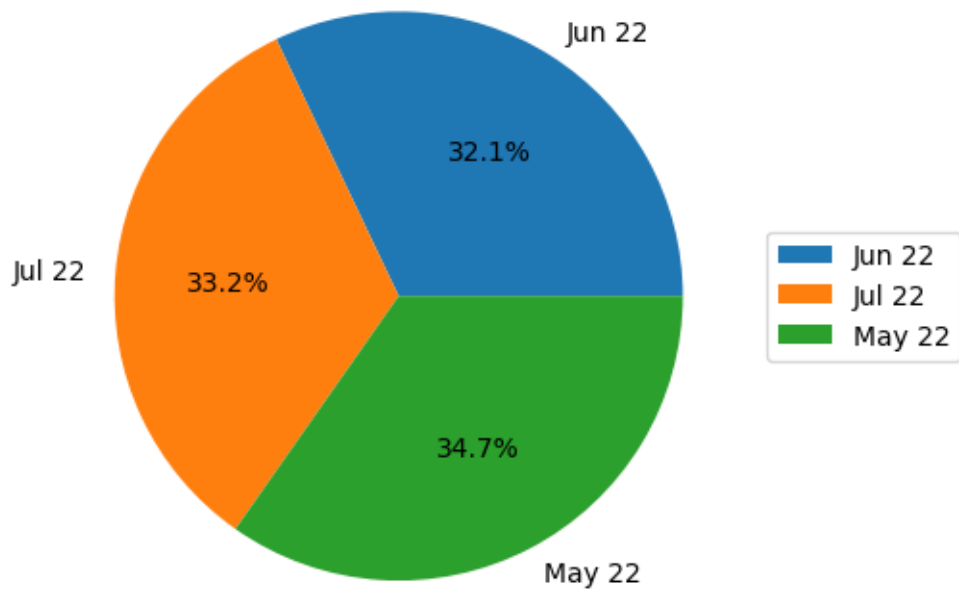
# Add a legend
ax.legend(grouped_data.index, loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))

# Add a tooltip
tooltip = mpld3.plugins.PointLabelTooltip(ax.patches, labels=grouped_data.index)
mpld3.plugins.connect(fig, tooltip)

# Remove x-axis and y-axis titles
ax.set_xlabel('')
ax.set_ylabel('')
ax.set_title('Month by month revenue')

plt.show()
```

Month by month revenue



#### g- Revenue by hotel Type

```
[81]: df_bookings.sample()
```

```
[81]:      booking_id  property_id booking_date check_in_date \
87174  Jun292217559RT214      17559    23-06-22    29-06-22

      checkout_date  no_guests room_category booking_platform ratings_given \
87174    30-06-22        2.0         RT2        logtrip          NaN

      booking_status  revenue_generated  revenue_realized
87174    Cancelled              15300              6120
```

```
[82]: df_hotels.sample()
```

```
[82]:      property_id property_name category      city
18      19558  Atliq Grands   Luxury  Bangalore
```

```
[83]: df_agg_bookings.sample()
```

```
[83]:      property_id check_in_date room_category  successful_bookings  capacity \
5152      19561    21-Jun-22         RT3              11      29.0
```

```
occ_pct
5152    37.93
```

```
[85]: df_full_bookings.sample()
```

```
[85]:      booking_id  property_id booking_date check_in_date \
37036  Jun132216559RT311          16559    12/6/2022    2022-06-13

      checkout_date  no_guests room_category booking_platform ratings_given \
37036      19-06-22         2.0          RT3          others         NaN

      booking_status  revenue_generated  revenue_realized  property_name \
37036      Cancelled              20400              8160  Atliq Exotica

      category  city      date  mmm yy week no  day_type
37036   Luxury  Mumbai 2022-06-13  Jun 22    W 25  weekday
```

```
[87]: df_full_bookings.groupby('room_category').revenue_realized.sum().sort_values()
```

```
[87]: room_category
RT1    212879394
RT4    258867172
RT3    318622920
RT2    385138296
Name: revenue_realized, dtype: int64
```

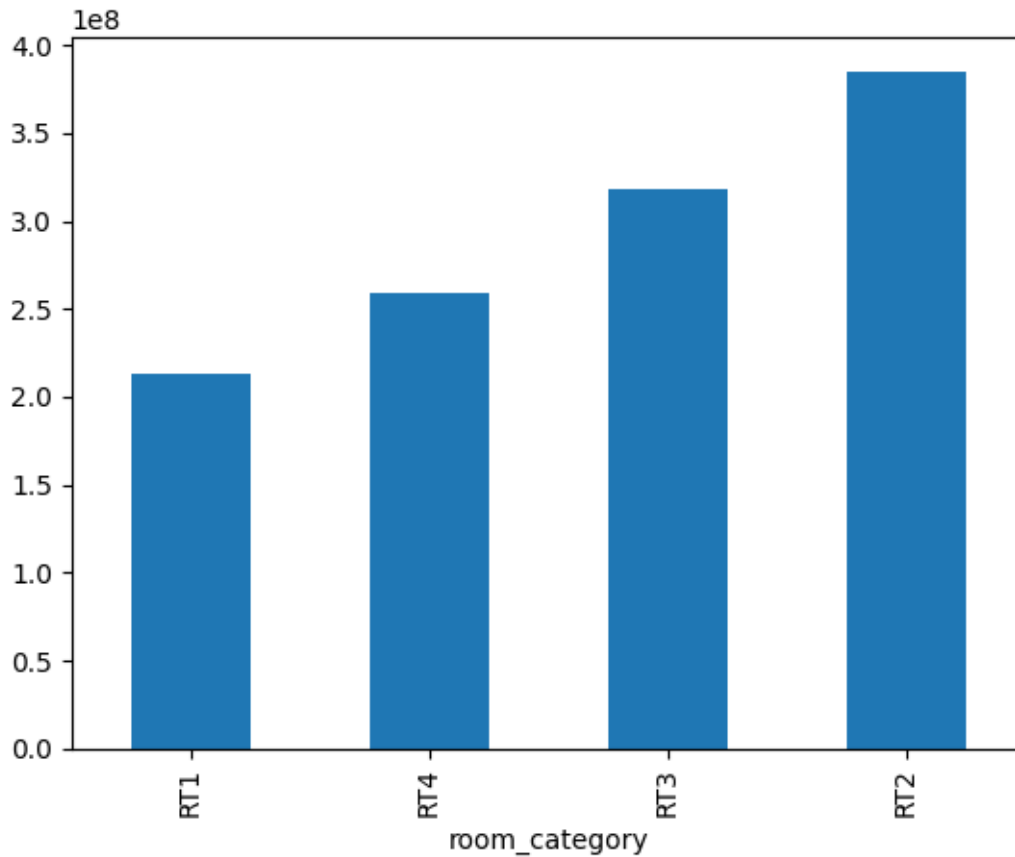
```
[89]: df_full_bookings.groupby('room_category').revenue_realized.sum().sort_values().
      ↪apply(amount_to_million)
```

```
[89]: room_category
RT1    212.88 M
RT4    258.87 M
RT3    318.62 M
RT2    385.14 M
Name: revenue_realized, dtype: object
```

```
[91]: df_full_bookings.groupby('room_category').revenue_realized.sum().sort_values().
      ↪plot(kind='bar')
```

```
[91]: <Axes: xlabel='room_category'>
```





```
[93]: import matplotlib

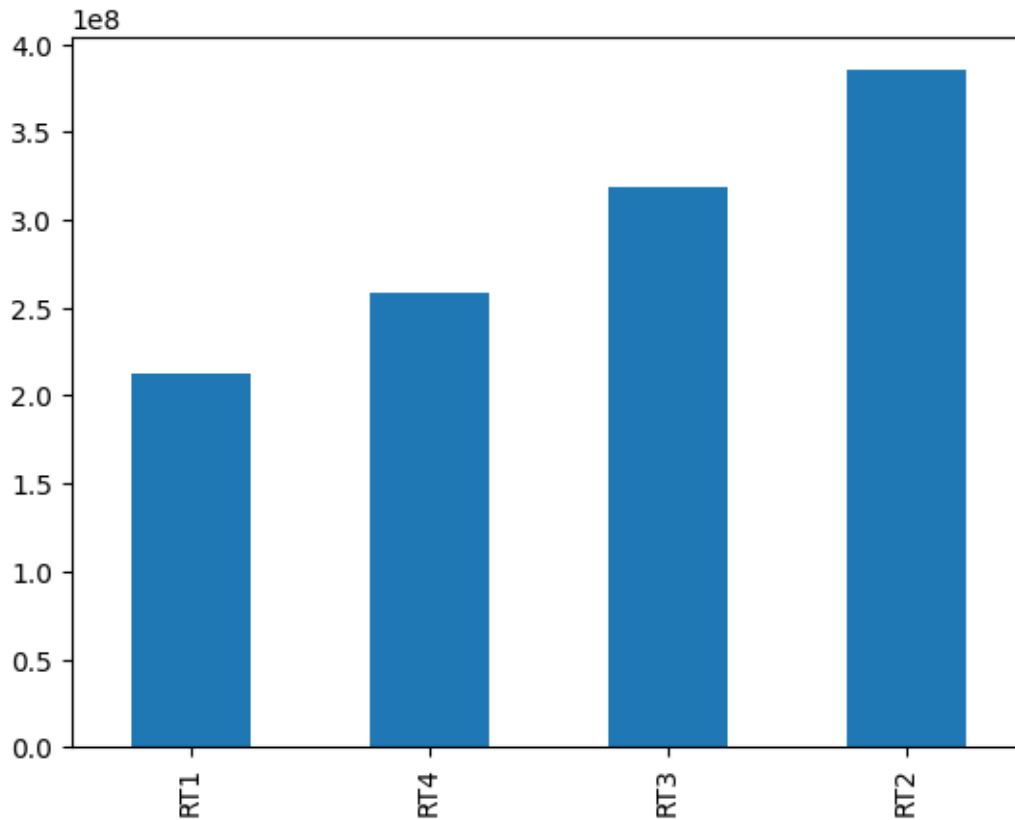
# Group the data and calculate the sum of revenue_realized
grouped_data = df_full_bookings.groupby('room_category').revenue_realized.sum().
    ↪sort_values()

# Create a bar plot with pandas
ax = grouped_data.plot(kind='bar')

# Add tooltips to the plot
matplotlib.cursor(hover=True)

# Remove x-axis and y-axis titles
ax.set_xlabel('')
ax.set_ylabel('')

plt.show()
```



```
[98]: # Group the data and calculate the sum of revenue_realized
grouped_data = df_full_bookings.groupby('room_category').revenue_realized.sum().
    ↪sort_values().apply(amount_to_million)

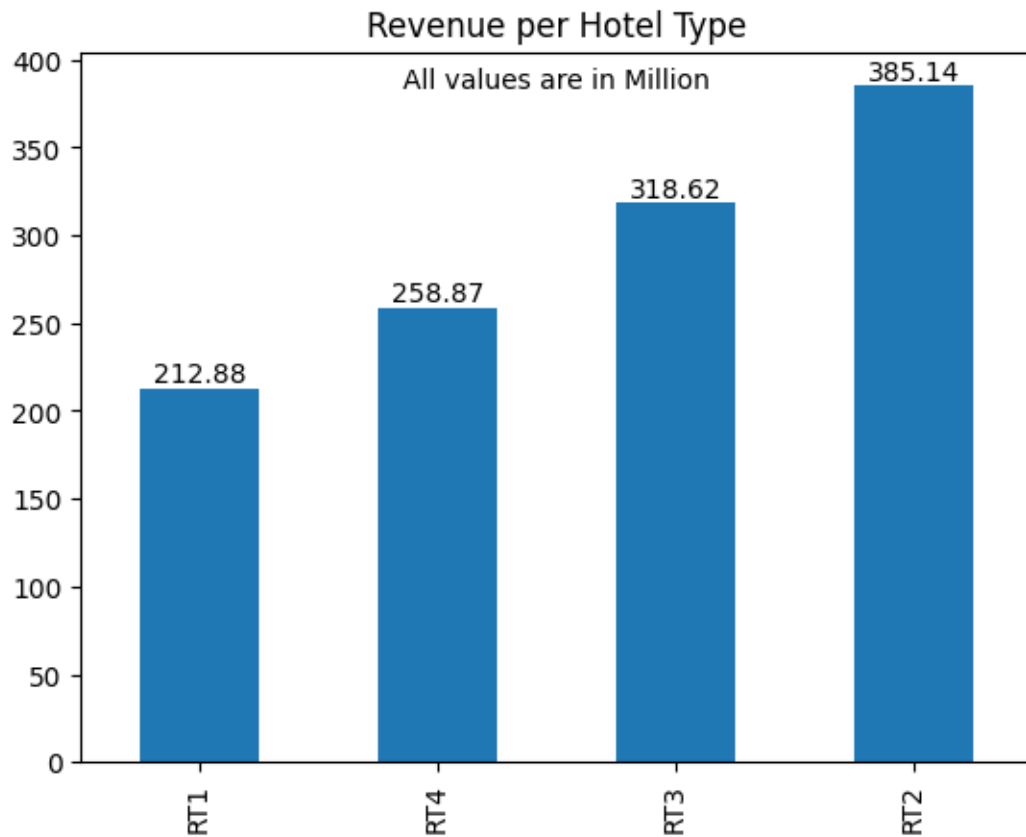
# Create a bar plot with pandas
ax = grouped_data.plot(kind='bar')

# Add value labels to the bars
for i, v in enumerate(grouped_data.values):
    ax.text(i, v, f'{v}', ha='center', va='bottom')

# Add a comment to the plot
ax.text(0.5, 0.95, 'All values are in Million', transform=ax.transAxes,
    ↪fontsize=10, ha='center')

# Remove x-axis and y-axis titles
ax.set_xlabel('')
ax.set_ylabel('')
ax.set_title('Revenue per Hotel Type')
```

```
plt.show()
```



#### h- Average rating by City

```
[99]: df_full_bookings.sample()
```

```
[99]:      booking_id  property_id booking_date check_in_date \
77693  Jul202219559RT110      19559      19-07-22      2022-07-20

      checkout_date  no_guests room_category booking_platform ratings_given \
77693      21-07-22         3.0          RT1      direct online          NaN

      booking_status  revenue_generated  revenue_realized  property_name \
77693      Cancelled              10725              4290  Atliq Exotica

      category      city      date  mmm yy week no  day_type
77693  Luxury  Bangalore  2022-07-20  Jul 22   W 30  weekday
```

```
[100]: df_full_bookings.groupby('city').ratings_given.mean()
```

```
[100]: city
      Bangalore    3.403911
      Delhi        3.775088
      Hyderabad    3.664286
      Mumbai       3.644350
      Name: ratings_given, dtype: float64
```

#### i- Pie chart of revenue realized per booking platform

```
[103]: df_full_bookings.groupby('booking_platform').revenue_realized.sum().
      ↪ apply(amount_to_million)
```

```
[103]: booking_platform
      direct offline    59.30
      direct online    117.25
      journey          71.23
      logtrip          129.04
      makeyourtrip     233.13
      others           480.70
      tripster         84.87
      Name: revenue_realized, dtype: float64
```

```
[110]: # Group the data and calculate the sum of revenue_realized
grouped_data = df_full_bookings.groupby('booking_platform').revenue_realized.
      ↪ sum().sort_values()

# Create a pie chart with matplotlib
fig, ax = plt.subplots()
ax.pie(grouped_data.values, labels=grouped_data.index, autopct='%1.1f%%')

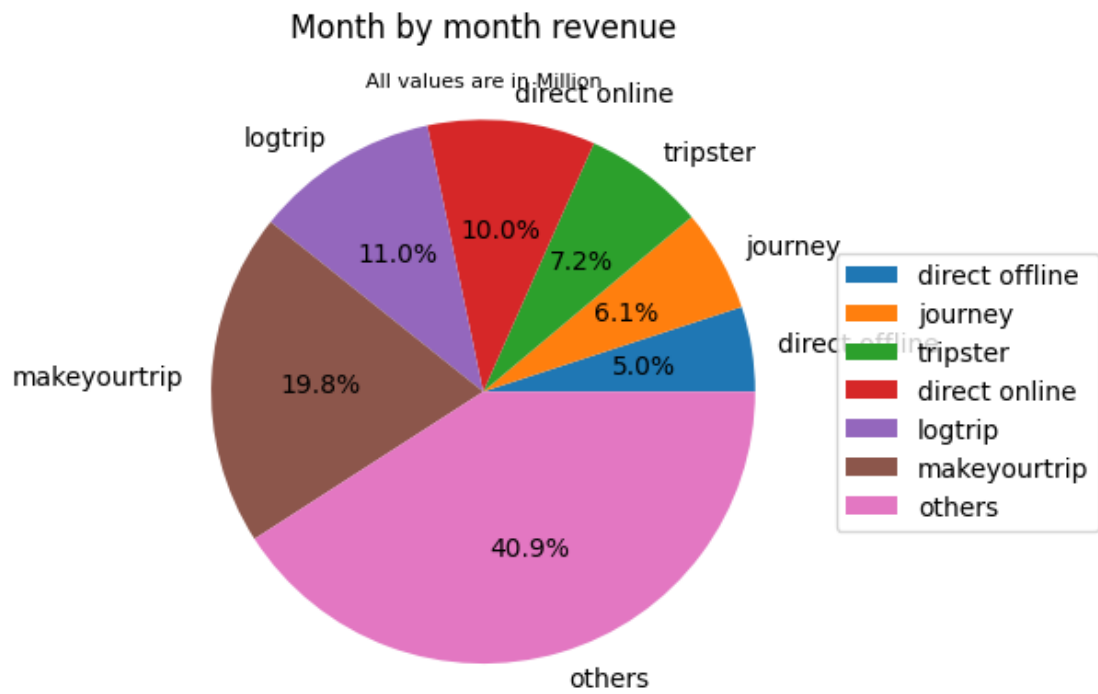
# Add a legend
ax.legend(grouped_data.index, loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))

# Add a tooltip
tooltip = mpld3.plugins.PointLabelTooltip(ax.patches, labels=grouped_data.index)
mpld3.plugins.connect(fig, tooltip)

ax.text(0.5, 0.95, 'All values are in Million', transform=ax.transAxes,
      ↪ fontsize=8, ha='center')

# Remove x-axis and y-axis titles
ax.set_xlabel('')
ax.set_ylabel('')
ax.set_title('Month by month revenue')

plt.show()
```



[ ]: