jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

AtliQ Hotels Data Analysis Project

[2]:
```
import pandas as pd
```

==> 1. Data Import and Data Exploration

Datasets

We have 5 csv file

- dim_date.csv
- dim_hotels.csv
- dim_rooms.csv
- fact_aggregated_bookings
- fact_bookings.csv

Read bookings data in a datagrame

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

Explore bookings data

[4]:
```
df_bookings.head()
```

[4]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue
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	

[5]:
```
df_bookings.shape
```

[5]:
```
(134590, 12)
```

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

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

[6]:
```
array(['RT1', 'RT2', 'RT3', 'RT4'], dtype=object)
```

[7]:
```
df_bookings.booking_platform.unique()
```

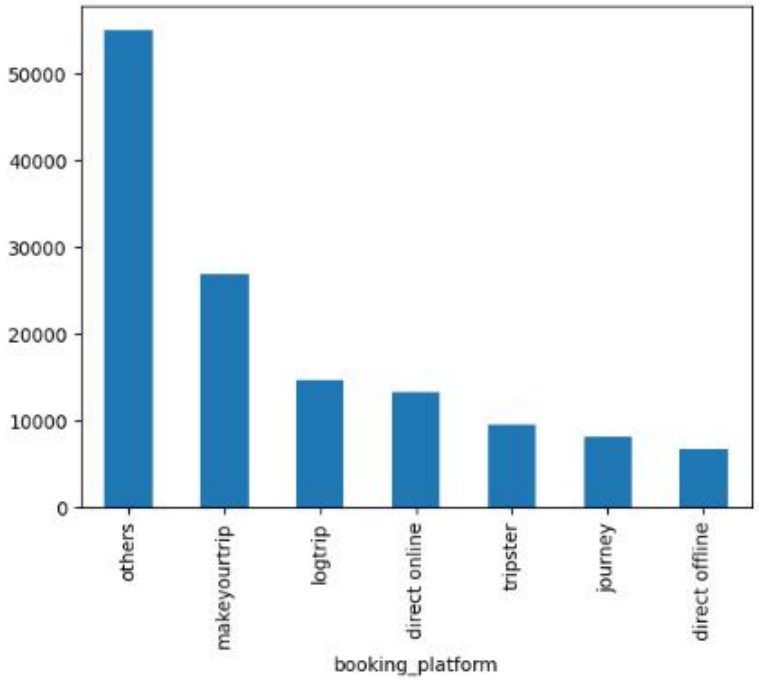
[7]:
```
array(['direct online', 'others', 'logtrip', 'tripster', 'makeyourtrip',
       'journey', 'direct offline'], dtype=object)
```

[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()
```

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

booking_platform

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

[10]:

	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

Read rest of the files

[11]:
```
df_date = pd.read_csv('datasets/dim_date.csv')
df_hotels = pd.read_csv('datasets/dim_hotels.csv')
df_rooms = pd.read_csv('datasets/dim_rooms.csv')
df_agg_bookings = pd.read_csv('datasets/fact_aggregated_bookings.csv')
```

[12]:
```
df_hotels.shape
```

[12]:
```
(25, 4)
```

[13]:
```
df_hotels.head(3)
```

[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

[14]:
```
df_hotels.category.value_counts()
```

[14]:
```
category
Luxury      16
Business     9
Name: count, dtype: int64
```

[15]:
```
df_hotels.city.value_counts().plot(kind="bar")
```

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

jupyter hotels_analysis Last Checkpoint: last year

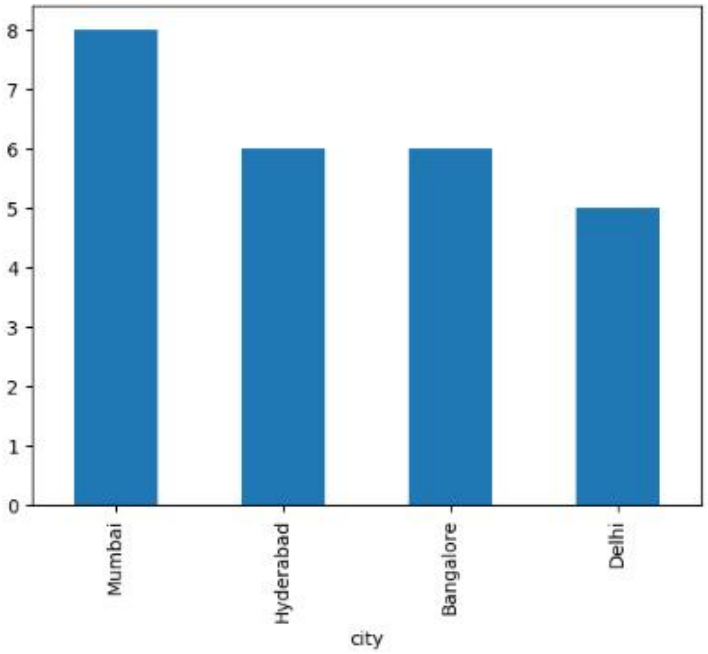
File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

Name: count, dtype: int64

[15]:
```
df_hotels.city.value_counts().plot(kind="bar")
```

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



Exercise: Explore aggregate bookings

---

[16]:
```
df_agg_bookings.head(3)
```

[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

Exercise-1. Find out unique property ids in aggregate bookings dataset

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

[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])
```

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

Exercise-1. Find out unique property ids in aggregate bookings dataset

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

[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])
```

Exercise-2. Find out total bookings per property_id

[18]:
```
# write your code here
df_agg_bookings.groupby('property_id')['successful_bookings'].sum()
```

[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
17564    3982
18558    4475
18559    5256
18560    6638
18561    6458
18562    7333
18563    4737
19558    4400
19559    4729
19560    6079
19561    5736
19562    5812
19563    5413
Name: successful_bookings, dtype: int64
```

Exercise-3. Find out days on which bookings are greater than capacity

[19]:
```
# write your code here
df_agg_bookings[df_agg_bookings.successful_bookings > df_agg_bookings.capacity]
```

[19]:

	property_id	check_in_date	room_category	successful_bookings	capacity
3	17558	1-May-22	RT1	30	19.0
12	16563	1-May-22	RT1	100	41.0
4136	19558	11-Jun-22	RT2	50	39.0
6209	19560	2-Jul-22	RT1	123	26.0
8522	19559	25-Jul-22	RT1	35	24.0

Exercise-3. Find out days on which bookings are greater than capacity

[19]:
```
# write your code here
df_agg_bookings[df_agg_bookings.successful_bookings > df_agg_bookings.capacity]
```

[19]:

	property_id	check_in_date	room_category	successful_bookings	capacity
3	17558	1-May-22	RT1	30	19.0
12	16563	1-May-22	RT1	100	41.0
4136	19558	11-Jun-22	RT2	50	39.0
6209	19560	2-Jul-22	RT1	123	26.0
8522	19559	25-Jul-22	RT1	35	24.0
9194	18563	31-Jul-22	RT4	20	18.0

Exercise-4. Find out properties that have highest capacity

[20]:
```
# write your code here
df_agg_bookings.capacity.max()
```

[20]:
```
np.float64(50.0)
```

==> 2. Data Cleaning

---

[21]:
```
df_bookings.describe()
```

[21]:

	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

(1) Clean invalid guests

[22]:
```
df_bookings[df_bookings.no_guests<=0]
```

==> 2. Data Cleaning

[21]:
```
df_bookings.describe()
```

[21]:

	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

(1) Clean invalid guests

[22]:
```
df_bookings[df_bookings.no_guests<=0]
```

[22]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	r
0	May012216558RT11	16558	27-04-22	1/5/2022	2/5/2022	-3.0	RT1	direct online	1.0	Checked Out	
3	May012216558RT14	16558	28-04-22	1/5/2022	2/5/2022	-2.0	RT1	others	NaN	Cancelled	
17924	May122218559RT44	18559	12/5/2022	12/5/2022	14-05-22	-10.0	RT4	direct online	NaN	No Show	
18020	May122218561RT22	18561	8/5/2022	12/5/2022	14-05-22	-12.0	RT2	makeyourtrip	NaN	Cancelled	
18119	May122218562RT311	18562	5/5/2022	12/5/2022	17-05-22	-6.0	RT3	direct offline	5.0	Checked Out	
18121	May122218562RT313	18562	10/5/2022	12/5/2022	17-05-22	-4.0	RT3	direct online	NaN	Cancelled	
56715	Jun082218562RT12	18562	5/6/2022	8/6/2022	13-06-22	-17.0	RT1	others	NaN	Checked Out	
119765	Jul202219560RT220	19560	19-07-22	20-07-22	22-07-22	-1.0	RT2	others	NaN	Checked Out	
134586	Jul312217564RT47	17564	30-07-22	31-07-22	1/8/2022	-4.0	RT4	logtrip	2.0	Checked Out	

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

[23]:
```
df_bookings = df_bookings[df_bookings.no_guests>0]
```

[24]:
```
df_bookings.shape
```

[24]:
```
(134578, 12)
```

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

(2) Outlier removal in revenue generated

[25]:
```
df_bookings.revenue_generated.min(), df_bookings.revenue_generated.max()
```

[25]:
```
(np.int64(6500), np.int64(28560000))
```

[26]:
```
df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.median()
```

[26]:
```
(np.float64(15378.036937686695), np.float64(13500.0))
```

[27]:
```
avg, std = df_bookings.revenue_generated.mean(), df_bookings.revenue_generated.std()
```

[28]:
```
higher_limit = avg + 3*std
higher_limit
```

[28]:
```
np.float64(294498.50173207896)
```

[29]:
```
lower_limit = avg - 3*std
lower_limit
```

[29]:
```
np.float64(-263742.4278567056)
```

[30]:
```
df_bookings[df_bookings.revenue_generated<=0]
```

[30]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_generat

[31]:
```
df_bookings[df_bookings.revenue_generated>higher_limit]
```

[31]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status
2	May012216558RT13	16558	28-04-22	1/5/2022	4/5/2022	2.0	RT1	logtrip	5.0	Checked Out
111	May012216559RT32	16559	29-04-22	1/5/2022	2/5/2022	6.0	RT3	direct online	NaN	Checked Out
315	May012216562RT22	16562	28-04-22	1/5/2022	4/5/2022	2.0	RT2	direct offline	3.0	Checked Out
562	May012217559RT118	17559	26-04-22	1/5/2022	2/5/2022	2.0	RT1	others	NaN	Cancelled
129176	Jul282216562RT26	16562	21-07-22	28-07-22	29-07-22	2.0	RT2	direct online	3.0	Checked Out

[32]:
```
df_bookings = df_bookings[df_bookings.revenue_generated<=higher_limit]
df_bookings.shape
```

[32]:
```
(134573, 12)
```

[33]:
```
df_bookings.revenue_realized.describe()
```

[33]:
```
count    134573.000000
mean      12695.983585
```

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

[33]:
```
df_bookings.revenue_realized.describe()
```

[33]:
```
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
```

[34]:
```
higher_limit = df_bookings.revenue_realized.mean() + 3*df_bookings.revenue_realized.std()
higher_limit
```

[34]:
```
np.float64(33479.358661845814)
```

[35]:
```
df_bookings[df_bookings.revenue_realized>higher_limit]
```

[35]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	r
137	May012216559RT41	16559	27-04-22	1/5/2022	7/5/2022	4.0	RT4	others	NaN	Checked Out	
139	May012216559RT43	16559	1/5/2022	1/5/2022	2/5/2022	6.0	RT4	tripster	3.0	Checked Out	
143	May012216559RT47	16559	28-04-22	1/5/2022	3/5/2022	3.0	RT4	others	5.0	Checked Out	
149	May012216559RT413	16559	24-04-22	1/5/2022	7/5/2022	5.0	RT4	logtrip	NaN	Checked Out	
222	May012216560RT45	16560	30-04-22	1/5/2022	3/5/2022	5.0	RT4	others	3.0	Checked Out	
...	...	...	...	...	...	...	...	...	...	...	
134328	Jul312219560RT49	19560	31-07-22	31-07-22	2/8/2022	6.0	RT4	direct online	5.0	Checked Out	
134331	Jul312219560RT412	19560	31-07-22	31-07-22	1/8/2022	6.0	RT4	others	2.0	Checked Out	
134467	Jul312219562RT45	19562	28-07-22	31-07-22	1/8/2022	6.0	RT4	makeyourtrip	4.0	Checked Out	
134474	Jul312219562RT412	19562	25-07-22	31-07-22	6/8/2022	5.0	RT4	direct offline	5.0	Checked Out	
134581	Jul312217564RT42	17564	31-07-22	31-07-22	1/8/2022	4.0	RT4	makeyourtrip	4.0	Checked Out	

1299 rows × 12 columns

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

[36]:
```
df_bookings[df_bookings.room_category=="RT4"].revenue_realized.describe()
```

[36]:
```
count    16071.000000
mean     23439.308444
std       9048.599076
min       7600.000000
```

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help

Markdown JupyterLab Python 3 (ipykernel)

[37]:
```
# mean + 3*standard deviation
23439+3*9048
```

[37]:
```
50583
```

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

[38]:
```
df_bookings[df_bookings.booking_id=="May012216558RT213"]
```

[38]:

booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_generat

[39]:
```
df_bookings.isnull().sum()
```

[39]:
```
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

[ ]:

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)

[40]:
```
# write your code here
df_agg_bookings.isnull().sum()
```

[40]:
```
property_id           0
check_in_date         0
room_category         0
successful_bookings   0
capacity              2
dtype: int64
```

[41]:
```
df_agg_bookings[df_agg_bookings.capacity.isnull()]
```

[41]:

	property_id	check_in_date	room_category	successful_bookings	capacity
8	17561	1-May-22	RT1	22	NaN

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

[43]:
```
df_agg_bookings.loc[[8,14]]
```

[43]:

	property_id	check_in_date	room_category	successful_bookings	capacity
8	17561	1-May-22	RT1	22	25.0
14	17562	1-May-22	RT1	12	25.0

Exercise-2. In aggregate bookings find out records that have successful_bookings value greater than capacity. Filter those records

[44]:
```
# write your code here
df_agg_bookings[df_agg_bookings['successful_bookings'] > df_agg_bookings['capacity']]
```

[44]:

	property_id	check_in_date	room_category	successful_bookings	capacity
3	17558	1-May-22	RT1	30	19.0
12	16563	1-May-22	RT1	100	41.0
4136	19558	11-Jun-22	RT2	50	39.0
6209	19560	2-Jul-22	RT1	123	26.0
8522	19559	25-Jul-22	RT1	35	24.0
9194	18563	31-Jul-22	RT4	20	18.0

==> 3. Data Transformation

Create occupancy percentage column

[45]:
```
df_agg_bookings.head(3)
```

[45]:

	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

[46]:
```
df_agg_bookings['occ_pct'] = df_agg_bookings.apply(lambda row: row['successful_bookings']/row['capacity'], axis=1)
```

You can use following approach to get rid of SettingWithCopyWarning

[47]:
```
new_col = df_agg_bookings.apply(lambda row: row['successful_bookings']/row['capacity'], axis=1)
df_agg_bookings = df_agg_bookings.assign(occ_pct=new_col.values)
df_agg_bookings.head(3)
```

[47]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

[47]:

	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

Convert it to a percentage value

[48]:
```
df_agg_bookings['occ_pct'] = df_agg_bookings['occ_pct'].apply(lambda x: round(x*100, 2))
df_agg_bookings.head(3)
```

[48]:

	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

[49]:
```
df_bookings.head()
```

[49]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue
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	

[50]:
```
df_agg_bookings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9200 entries, 0 to 9199
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   property_id          9200 non-null   int64
 1   check_in_date        9200 non-null   object
 2   room_category        9200 non-null   object
 3   successful_bookings  9200 non-null   int64
 4   capacity             9200 non-null   float64
 5   occ_pct              9200 non-null   float64
dtypes: float64(2), int64(2), object(2)
memory usage: 431.4+ KB
```

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

1. Creating new columns
2. Normalization

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help

Markdown JupyterLab Python 3 (ipykernel)

==> 4. Insights Generation

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

[51]:
```
df_agg_bookings.head(3)
```

[51]:

	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

[52]:
```
df_agg_bookings.groupby("room_category")["occ_pct"].mean()
```

[52]:
```
room_category
RT1    58.232748
RT2    58.040278
RT3    58.028213
RT4    59.300461
Name: occ_pct, dtype: float64
```

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

[53]:
```
df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_on="room_id")
df.head(4)
```

[53]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_id	room_class
0	16559	1-May-22	RT1	25	30.0	83.33	RT1	Standard
1	19562	1-May-22	RT1	28	30.0	93.33	RT1	Standard
2	19563	1-May-22	RT1	23	30.0	76.67	RT1	Standard
3	17558	1-May-22	RT1	30	19.0	157.89	RT1	Standard

[54]:
```
df.drop("room_id",axis=1, inplace=True)
df.head(4)
```

[54]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class
0	16559	1-May-22	RT1	25	30.0	83.33	Standard
1	19562	1-May-22	RT1	28	30.0	93.33	Standard
2	19563	1-May-22	RT1	23	30.0	76.67	Standard
3	17558	1-May-22	RT1	30	19.0	157.89	Standard

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

[55]:
```
df.groupby("room_class")["occ_pct"].mean()
```

[55]:
```
room_class
Elite           58.040278
Premium         58.028213
Presidential    59.300461
Standard        58.232748
Name: occ_pct, dtype: float64
```

[56]:
```
df[df.room_class=="Standard"].occ_pct.mean()
```

[56]:
```
np.float64(58.23274782608696)
```

2. Print average occupancy rate per city

[57]:
```
df_hotels.head(3)
```

[57]:

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

[58]:
```
df = pd.merge(df, df_hotels, on="property_id")
df.head(3)
```

[58]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city
0	16559	1-May-22	RT1	25	30.0	83.33	Standard	Atliq Exotica	Luxury	Mumbai
1	19562	1-May-22	RT1	28	30.0	93.33	Standard	Atliq Bay	Luxury	Bangalore
2	19563	1-May-22	RT1	23	30.0	76.67	Standard	Atliq Palace	Business	Bangalore

[59]:
```
df.groupby("city")["occ_pct"].mean()
```

[59]:
```
city
Bangalore    56.594207
Delhi        61.606467
Hyderabad    58.144651
Mumbai       57.943142
Name: occ_pct, dtype: float64
```

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

[60]:
```
df_date.head(3)
```

[60]:

	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

[60]:

	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

[61]:
```
df = pd.merge(df, df_date, left_on="check_in_date", right_on="date")
df.head(3)
```

[61]:

	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
0	19563	10-May-22	RT3	15	29.0	51.72	Premium	Atliq Palace	Business	Bangalore	10-May-22	May 22	W 20	weekeda
1	18560	10-May-22	RT1	19	30.0	63.33	Standard	Atliq City	Business	Hyderabad	10-May-22	May 22	W 20	weekeda
2	19562	10-May-22	RT1	18	30.0	60.00	Standard	Atliq Bay	Luxury	Bangalore	10-May-22	May 22	W 20	weekeda

[62]:
```
df.groupby("day_type")["occ_pct"].mean().round(2)
```

[62]:
```
day_type
weekeday    50.90
weekend     72.39
Name: occ_pct, dtype: float64
```

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

[63]:
```
df_june_22 = df[df["mmm yy"]=="Jun 22"]
df_june_22.head(4)
```

[63]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	date	mmm yy	week no	day_t
2200	16559	10-Jun-22	RT1	20	30.0	66.67	Standard	Atliq Exotica	Luxury	Mumbai	10-Jun-22	Jun 22	W 24	weeke
2201	19562	10-Jun-22	RT1	19	30.0	63.33	Standard	Atliq Bay	Luxury	Bangalore	10-Jun-22	Jun 22	W 24	weeke
2202	19563	10-Jun-22	RT1	17	30.0	56.67	Standard	Atliq Palace	Business	Bangalore	10-Jun-22	Jun 22	W 24	weeke

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help

Markdown JupyterLab Python 3 (ipykernel)

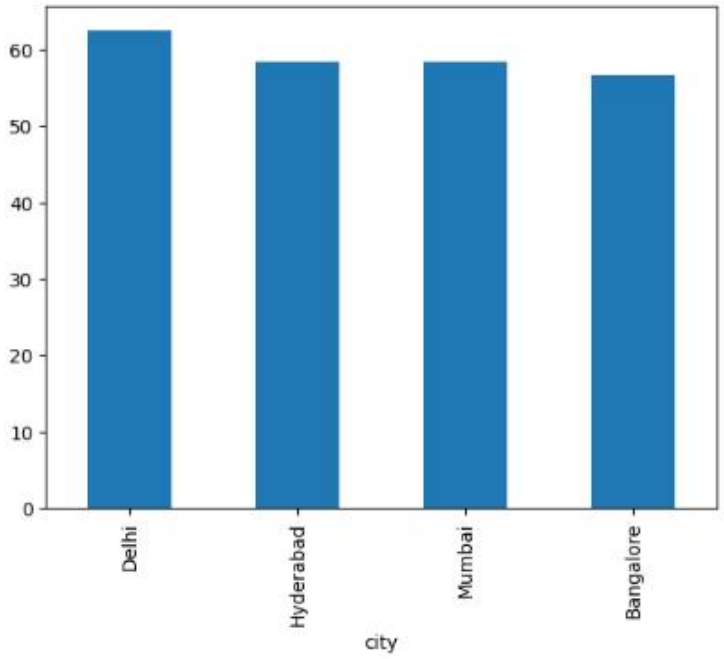
2201	19562	10-Jun-22	RT1	19	30.0	63.33	Standard	Atliq Bay	Luxury	Bangalore	10-Jun-22	Jun 22	W 24	weeke
2202	19563	10-Jun-22	RT1	17	30.0	56.67	Standard	Atliq Palace	Business	Bangalore	10-Jun-22	Jun 22	W 24	weeke
2203	17558	10-Jun-22	RT1	9	19.0	47.37	Standard	Atliq Grands	Luxury	Mumbai	10-Jun-22	Jun 22	W 24	weeke

[64]:
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).sort_values(ascending=False)
```

[64]:
```
city
Delhi        62.47
Hyderabad    58.46
Mumbai       58.38
Bangalore    56.58
Name: occ_pct, dtype: float64
```

[65]:
```
df_june_22.groupby('city')['occ_pct'].mean().round(2).sort_values(ascending=False).plot(kind="bar")
```

[65]:
```
<Axes: xlabel='city'>
```



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

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

[66]:
```
df_august = pd.read_csv("datasets/new_data_august.csv")
df_august.head(3)
```

[66]:

	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	100.00
1	19562	Atliq Bay	Luxury	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	21	30	70.00
2	19563	Atliq Palace	Business	Bangalore	RT1	Standard	01-Aug-22	Aug-22	W 32	weekeday	23	30	76.67

[67]:
```
df_august.columns
```

[67]:
```
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')
```

[68]:
```
df.columns
```

[68]:
```
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')
```

[69]:
```
df_august.shape
```

[69]:
```
(7, 13)
```

[70]:
```
df.shape
```

[70]:
```
(6500, 14)
```

[71]:
```
latest_df = pd.concat([df, df_august], ignore_index = True, axis = 0)
latest_df.tail(10)
```

[71]:

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class	property_name	category	city	date	mmm yy	week no	day_t
6497	17558	31-Jul-22	RT4	3	6.0	50.0	Presidential	Atliq Grands	Luxury	Mumbai	31-Jul-22	Jul 22	W 32	week
6498	19563	31-Jul-22	RT4	3	6.0	50.0	Presidential	Atliq Palace	Business	Bangalore	31-Jul-22	Jul 22	W 32	week
6499	17561	31-Jul-22	RT4	3	4.0	75.0	Presidential	Atliq Blu	Luxury	Mumbai	31-Jul-22	Jul 22	W 32	week
6500	16559	01-Aug-22	RT1	30	30.0	NaN	Standard	Atliq Exotica	Luxury	Mumbai	NaN	Aug-	W 32	week

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

[72]:
```
latest_df.shape
```

[72]:
```
(6507, 15)
```

Check this post for codebasics resume project challange winner entry: https://www.linkedin.com/posts/ashishbabaria_codebasicsresumeprojectchallenge-data-powerbi-activity-6977940034414886914-dmoJ?utm_source=share&utm_medium=member_desktop

6. Print revenue realized per city

[73]:
```
df_bookings.head()
```

[73]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue
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	

[74]:
```
df_hotels.head(3)
```

[74]:

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

[75]:
```
df_bookings_all = pd.merge(df_bookings, df_hotels, on="property_id")
df_bookings_all.head(3)
```

[75]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue
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	

[76]:
```
df_bookings_all.groupby("city")["revenue_realized"].sum()
```

[76]:
```
city
Bangalore    420383550
Delhi        294404488
```

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Markdown JupyterLab Python 3 (ipykernel)

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

[76]:
```
df_bookings_all.groupby("city")["revenue_realized"].sum()
```

[76]:
```
city
Bangalore    420383550
Delhi        294404488
Hyderabad    325179310
Mumbai       668569251
Name: revenue_realized, dtype: int64
```

7. Print month by month revenue

[77]:
```
df_date.head(3)
```

[77]:

	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

[78]:
```
df_date["mmm yy"].unique()
```

[78]:
```
array(['May 22', 'Jun 22', 'Jul 22'], dtype=object)
```

[79]:
```
df_bookings_all.head(3)
```

[79]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue
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	

[80]:
```
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)
```

Jupyter hotels_analysis Last Checkpoint: last year

Edit View Run Kernel Settings Help

Code JupyterLab Python 3 (ipykernel)

[81]:

	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

[82]:
```
df_bookings_all.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
```

[ ]:
```
df_bookings_all = pd.merge(df_bookings_all, df_date, left_on="check_in_date", right_on="date")
df_bookings_all.head(3)
```

[ ]:
```
df_bookings_all.groupby("mmm yy")["revenue_realized"].sum()
```

Exercise-1. Print revenue realized per hotel type

[95]:
```
# write your code here
df_new = pd.merge(df_hotels,df_bookings,on = "property_id")
df_new.groupby('category')['revenue_realized'].sum()
```

[95]:
```
category
Business     655967037
Luxury      1052569562
Name: revenue_realized, dtype: int64
```

[97]:
```
df_bookings
```

[97]:

	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

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help

Code JupyterLab Python 3 (ipykernel)

```
df_new = pd.merge(df_hotels,df_bookings,on = "property_id")
df_new.groupby('category')['revenue_realized'].sum()
```

[95]:
```
category
Business     655967037
Luxury      1052569562
Name: revenue_realized, dtype: int64
```

[97]:
```
df_bookings
```

[97]:

	booking_id	property_id	booking_date	check_in_date	checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	re
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	
...	...	...	...	...	...	...	...	...	...	...	
134584	Jul312217564RT45	17564	30-07-22	31-07-22	1/8/2022	2.0	RT4	others	2.0	Checked Out	
134585	Jul312217564RT46	17564	29-07-22	31-07-22	3/8/2022	1.0	RT4	makeyourtrip	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	

134573 rows × 12 columns

Exercise-2 Print average rating per city

[99]:
```
# write your code here
df_new.groupby('city')['ratings_given'].mean()
```

[99]:
```
city
Bangalore    3.407681
Delhi        3.779298
Hyderabad    3.661041
Mumbai       3.650545
Name: ratings_given, dtype: float64
```

Exercise-3 Print a pie chart of revenue realized per booking platform

[86]:
```
# write your code here
```

[107]:
```
df_new.groupby("booking_platform")['revenue_realized'].sum().plot(kind = "pie")
```

[107]:
```
<Axes: ylabel='revenue_realized'>
```

jupyter hotels_analysis Last Checkpoint: last year

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
df_new.groupby('city')['ratings_given'].mean()
```

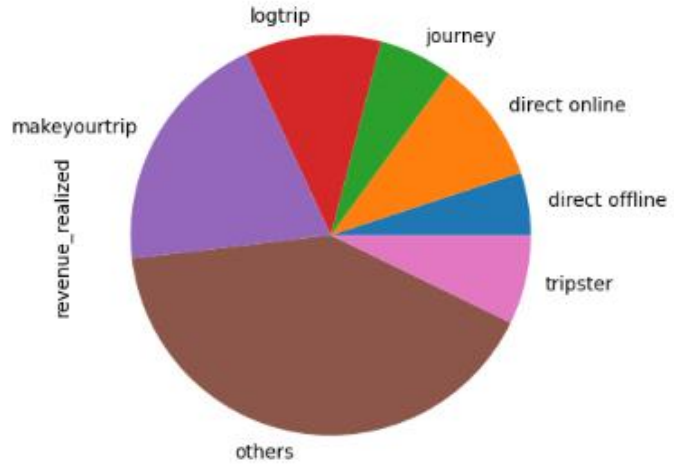
[99]:
```
city
Bangalore    3.407681
Delhi        3.779298
Hyderabad    3.661041
Mumbai       3.650545
Name: ratings_given, dtype: float64
```

Exercise-3 Print a pie chart of revenue realized per booking platform

[86]:
```
# write your code here
```

[107]:
```
df_new.groupby("booking_platform")['revenue_realized'].sum().plot(kind = "pie")
```

[107]:
```
<Axes: ylabel='revenue_realized'>
```



[ ]:

[ ]:

[ ]:

[ ]: