```
from google.colab import files
uploaded = files.upload()
```

Choose Files fact_bookings.csv

- **fact_bookings.csv**(text/csv) - 13048455 bytes, last modified: 10/10/2022 - 100% done

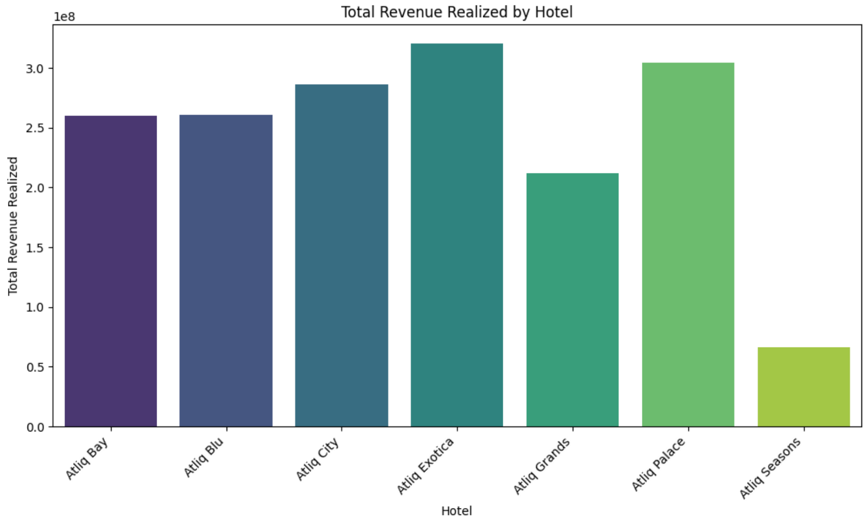
```
Saving fact_bookings.csv to fact_bookings.csv
```

```
import pandas as pd
dim_date = pd.read_csv('dim_date.csv')
dim_hotels = pd.read_csv('dim_hotels.csv')
dim_rooms = pd.read_csv('dim_rooms.csv')
fact_aggregated_bookings = pd.read_csv('fact_aggregated_bookings.csv')
fact_bookings = pd.read_csv('fact_bookings.csv')
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

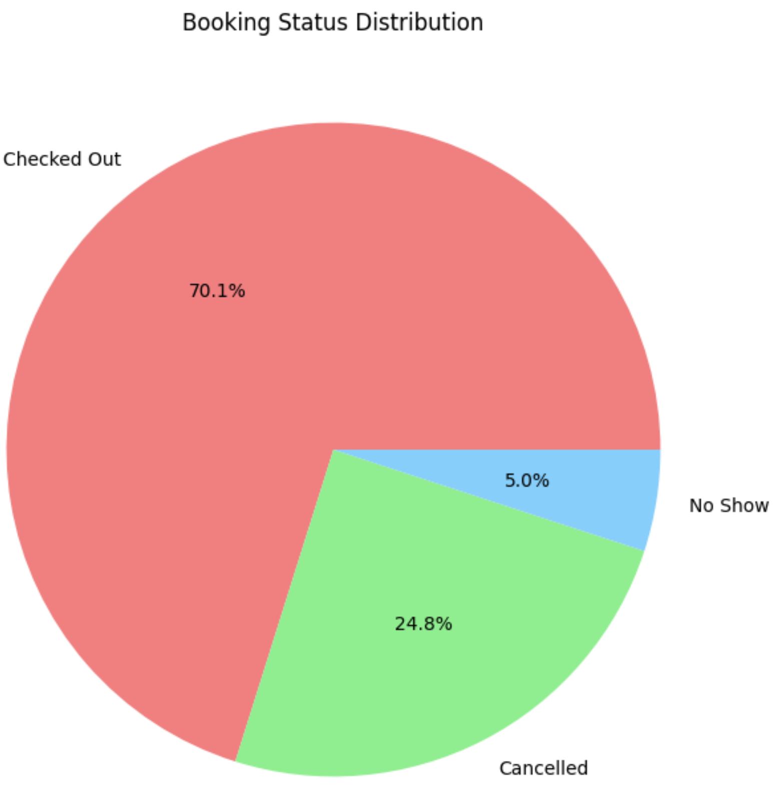
```
#Find the Sum of revenue_realized
total_revenue_realized = fact_bookings['revenue_realized'].sum()
print("Total Revenue Realized:", total_revenue_realized)

merged_data = pd.merge(fact_bookings, dim_hotels, on='property_id', how='inner')
total_revenue_by_hotel = merged_data.groupby('property_name')['revenue_realized'].sum().reset_index()

plt.figure(figsize=(12, 6))
sns.barplot(x='property_name', y='revenue_realized', data=total_revenue_by_hotel, palette='viridis')
plt.title('Total Revenue Realized by Hotel')
plt.xlabel('Hotel')
plt.ylabel('Total Revenue Realized')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
Total Revenue Realized: 1708771229
<ipython-input-46-0d3475a80391>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x='property_name', y='revenue_realized', data=total_revenue_by_hotel, p
```



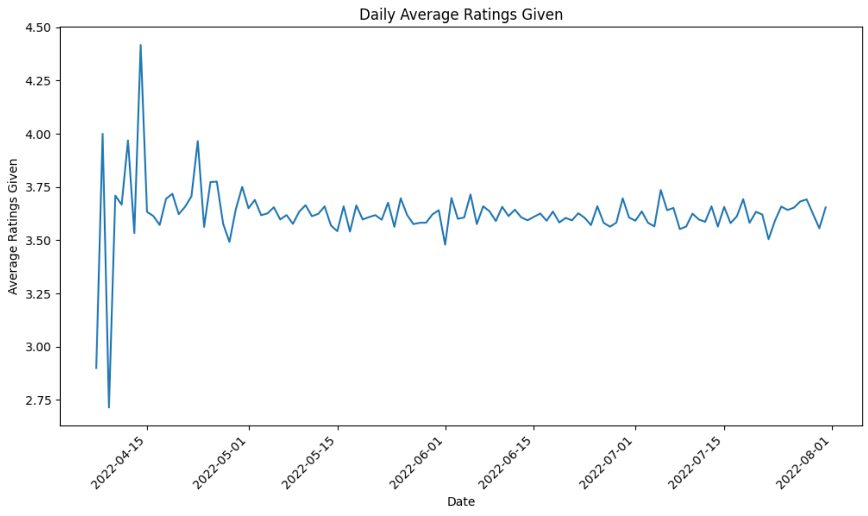
```
#Find the Total Number of Bookings using booking_id in fact_bookings
total_bookings = fact_bookings['booking_id'].nunique()
print("Total Number of Bookings:", total_bookings)


booking_status_distribution = fact_bookings['booking_status'].value_counts()

plt.figure(figsize=(8, 8))
plt.pie(booking_status_distribution, labels=booking_status_distribution.index, autopct='%1.1f%%', colors=['lightcoral', 'lightgreen', ':
plt.title('Booking Status Distribution')
plt.show()
```

```
Total Number of Bookings: 134590
```



```
#Find the Average of Ratings Given
average_ratings_given = fact_bookings['ratings_given'].mean()
print("Average Ratings Given:", average_ratings_given)

fact_bookings['booking_date'] = pd.to_datetime(fact_bookings['booking_date'])
daily_average_ratings = fact_bookings.groupby('booking_date')['ratings_given'].mean().reset_index()

plt.figure(figsize=(12, 6))
sns.lineplot(x='booking_date', y='ratings_given', data=daily_average_ratings)
plt.title('Daily Average Ratings Given')
plt.xlabel('Date')
plt.ylabel('Average Ratings Given')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
Average Ratings Given: 3.619003934160154
```

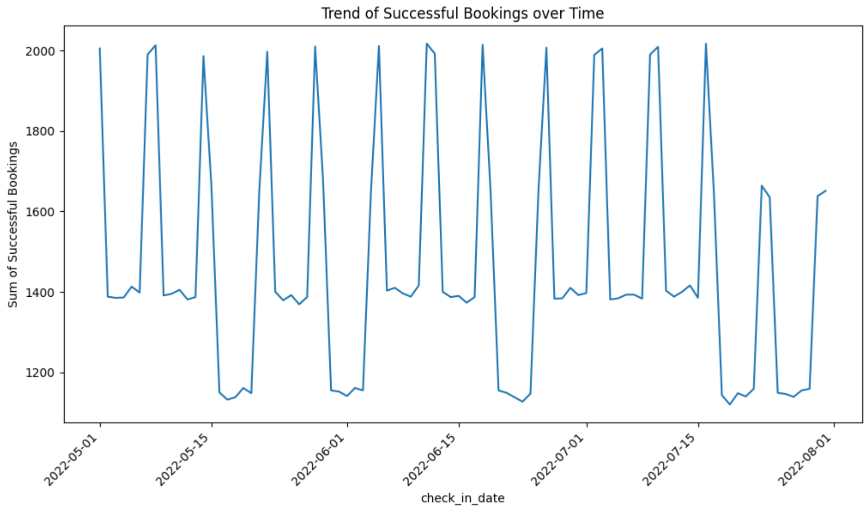


```
#Sum of Capacity
total_capacity = fact_aggregated_bookings['capacity'].sum()
print("Total Capacity:", total_capacity)
```

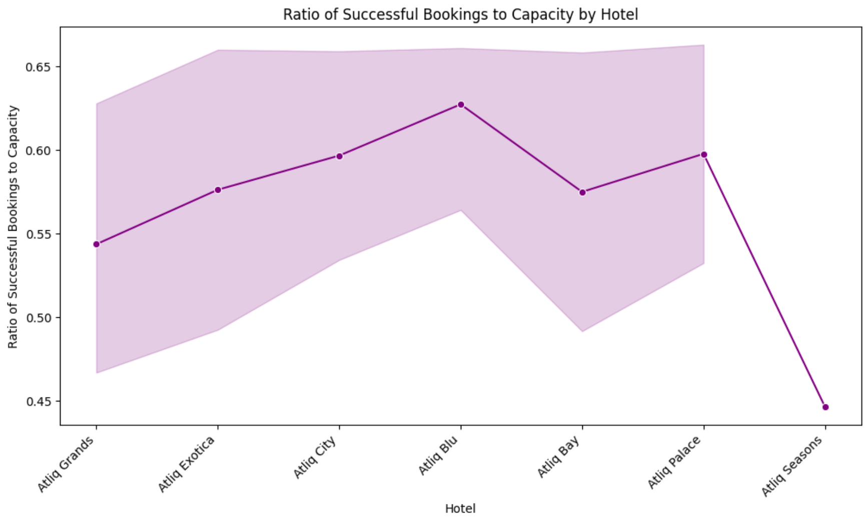
```
Total Capacity: 232576
```

```
#Sum of Successful Booking from fact_aggregated_bookings
total_successful_bookings = fact_aggregated_bookings['successful_bookings'].sum()
print("Total Successful Bookings:", total_successful_bookings)

plt.figure(figsize=(12, 6))
sns.lineplot(x='check_in_date', y='successful_bookings', data=sum_successful_bookings_over_time)
plt.title('Trend of Successful Bookings over Time')
plt.xlabel('check_in_date')
plt.ylabel('Sum of Successful Bookings')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
Total Successful Bookings: 134590
```



```
ratio_successful_to_capacity = fact_aggregated_bookings.groupby('property_id').agg({'successful_bookings': 'sum', 'capacity': 'sum'})
ratio_successful_to_capacity['ratio'] = ratio_successful_to_capacity['successful_bookings'] / ratio_successful_to_capacity['capacity']
ratio_successful_to_capacity = pd.merge(ratio_successful_to_capacity, dim_hotels, on='property_id', how='inner')

# Visualization: Line Plot - Ratio of Successful Bookings to Capacity by Hotel
plt.figure(figsize=(12, 6))
sns.lineplot(x='property_name', y='ratio', data=ratio_successful_to_capacity, marker='o', color='purple')
plt.title('Ratio of Successful Bookings to Capacity by Hotel')
plt.xlabel('Hotel')
plt.ylabel('Ratio of Successful Bookings to Capacity')
plt.xticks(rotation=45, ha='right')
plt.show()
```



```
#Count of booking_id where booking_status = "Cancelled"
cancelled_bookings_count = fact_bookings[fact_bookings['booking_status'] == 'Cancelled']['booking_id'].count()
print("Count of Cancelled Bookings:", cancelled_bookings_count)
```

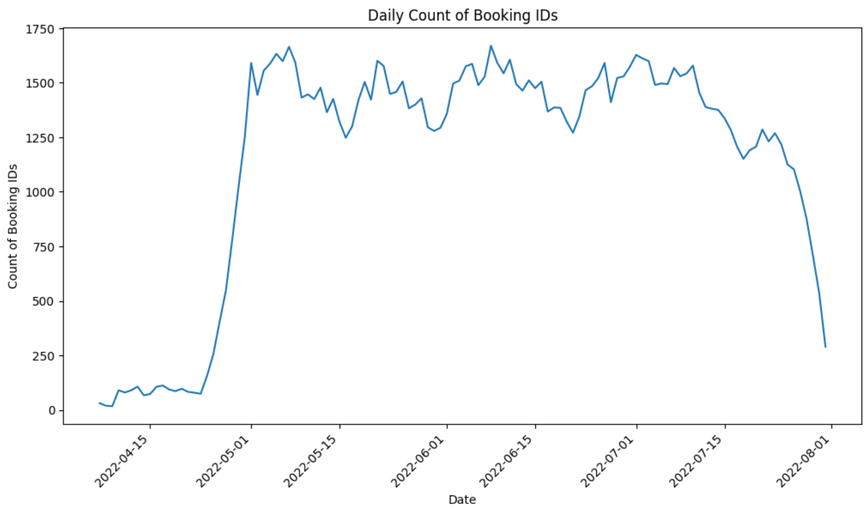
```
    Count of Cancelled Bookings: 33420
```

```
#Ratio of 'Total Cancelled Bookings' to 'Total Bookings'
ratio_cancelled_to_total_bookings = cancelled_bookings_count / total_bookings
print("Ratio of Total Cancelled Bookings to Total Bookings:", ratio_cancelled_to_total_bookings)
```

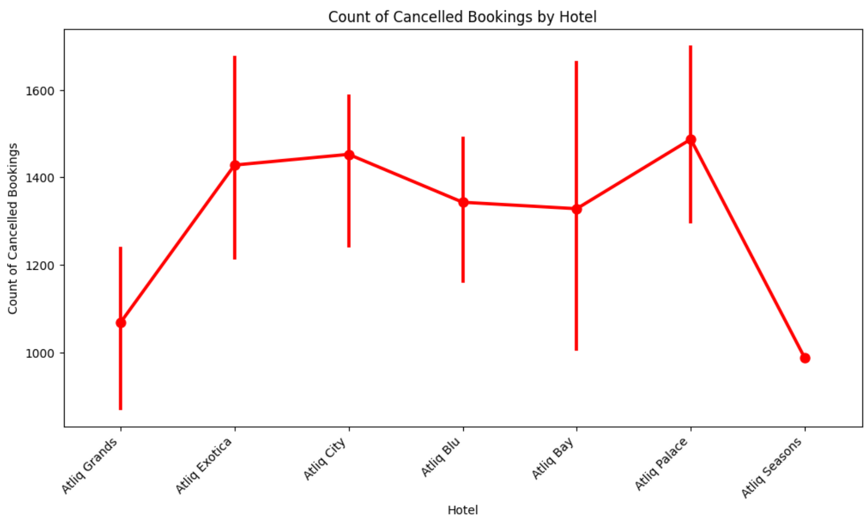
```
    Ratio of Total Cancelled Bookings to Total Bookings: 0.24830968125417935
```

```
booking_count_by_hotel = fact_bookings.groupby('property_id')['booking_id'].count().reset_index()
booking_count_by_hotel = pd.merge(booking_count_by_hotel, dim_hotels, on='property_id', how='inner')
```

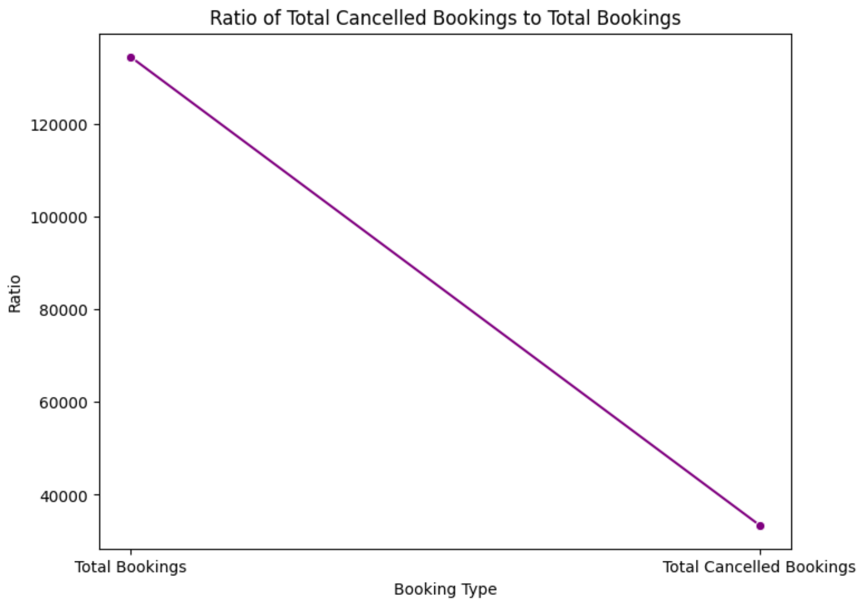
```
# Visualization 3: Line Plot - Daily Count of Booking IDs
fact_bookings['booking_date'] = pd.to_datetime(fact_bookings['booking_date'])
daily_booking_count = fact_bookings.groupby('booking_date')['booking_id'].count().reset_index()

plt.figure(figsize=(12, 6))
sns.lineplot(x='booking_date', y='booking_id', data=daily_booking_count)
plt.title('Daily Count of Booking IDs')
plt.xlabel('Date')
plt.ylabel('Count of Booking IDs')
plt.xticks(rotation=45, ha='right')
plt.show()
```



```
cancelled_bookings = fact_bookings[fact_bookings['booking_status'] == 'Cancelled']

# Visualization: Point Plot - Count of Cancelled Bookings by Hotel
cancelled_bookings_by_hotel = cancelled_bookings.groupby('property_id')['booking_id'].count().reset_index()
cancelled_bookings_by_hotel = pd.merge(cancelled_bookings_by_hotel, dim_hotels, on='property_id', how='inner')

plt.figure(figsize=(12, 6))
sns.pointplot(x='property_name', y='booking_id', data=cancelled_bookings_by_hotel, color='red', markers='o', linestyles='-', dodge=True
plt.title('Count of Cancelled Bookings by Hotel')
plt.xlabel('Hotel')
plt.ylabel('Count of Cancelled Bookings')
plt.xticks(rotation=45, ha='right')
plt.show()
```



```
total_bookings = fact_bookings['booking_id'].nunique()
total_cancelled_bookings = fact_bookings[fact_bookings['booking_status'] == 'Cancelled']['booking_id'].nunique()

# Create a DataFrame for visualization
ratio_data = pd.DataFrame({'Total Bookings': [total_bookings], 'Total Cancelled Bookings': [total_cancelled_bookings]})

# Visualization: Line Plot - Ratio of Total Cancelled Bookings to Total Bookings
plt.figure(figsize=(8, 6))
sns.lineplot(data=ratio_data.T.reset_index(), x='index', y=0, marker='o', color='purple')
plt.title('Ratio of Total Cancelled Bookings to Total Bookings')
plt.xlabel('Booking Type')
plt.ylabel('Ratio')
plt.show()
```



```
import ipywidgets as widgets
from IPython.display import display
```

```
property_dropdown = widgets.Dropdown(options=dim_hotels['property_name'].unique(), value=None, description='Property:')
city_dropdown = widgets.Dropdown(options=dim_hotels['city'].unique(), value=None, description='City:')
status_dropdown = widgets.Dropdown(options=fact_bookings['booking_status'].unique(), value=None, description='Status:')
platform_dropdown = widgets.Dropdown(options=fact_bookings['booking_platform'].unique(), value=None, description='Platform:')
month_dropdown = widgets.Dropdown(options=pd.to_datetime(fact_bookings['booking_date']).dt.month_name().unique(), value=None, description
#week_dropdown = widgets.Dropdown(options=fact_bookings['week no'].unique(), value=None, description='Week:')

# Function to filter data based on selected filters
def filter_data(property_name, city, status, platform, month):
    filtered_data = fact_bookings.copy()

    if property_name:
        filtered_data = filtered_data[filtered_data['property_name'] == property_name]
    if city:
        filtered_data = filtered_data[filtered_data['city'] == city]
    if status:
        filtered_data = filtered_data[filtered_data['booking_status'] == status]
    if platform:
        filtered_data = filtered_data[filtered_data['booking_platform'] == platform]
    if month:
        month_num = pd.to_datetime(filtered_data['booking_date']).dt.month_name() == month
        filtered_data = filtered_data[month_num]
    #if week:
        #filtered_data = filtered_data[filtered_data['week no'] == week]

    return filtered_data

# Update the data based on widget values
def update_data(change):
    filtered_data = filter_data(property_dropdown.value, city_dropdown.value, status_dropdown.value,
                                platform_dropdown.value, month_dropdown.value)
    # Display the filtered data or perform further analysis

# Attach the update_data function to widget events
property_dropdown.observe(update_data, names='value')
city_dropdown.observe(update_data, names='value')
status_dropdown.observe(update_data, names='value')
platform_dropdown.observe(update_data, names='value')
month_dropdown.observe(update_data, names='value')
#week_dropdown.observe(update_data, names='value')

# Display widgets
display(property_dropdown, city_dropdown, status_dropdown, platform_dropdown, month_dropdown)
```

```
 Property:
     City:
   Status:
 Platform:
    Month:
```

```
!jupyter nbconvert --to pdf /content/Hospitality_Analysis.ipynb
```

```
[NbConvertApp] WARNING | pattern '/content/Hospitality_Analysis.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
```