ALMABETTER

Hotel Booking Analysis EDA Project Using Python

Team Member

- Hariom Gupta
- Pallavi
- Aditya
- Vishwajeet
- Ritesh Verma

Project Summary

Hotel Booking Analysis project uses AlmaBetter EDA to analyze hotel reservations in various city and resort hotels. The dataset consists of 32 columns and 119390 rows, with data collection, cleansing, and manipulation divided into three categories. Column names, such as hotel, are updated as data collection progresses. Data manipulation is performed before visualizing data, examining null values and dropping columns with higher percentages. Data visualization is utilized to achieve better understanding and business goals.

Install & Import Libraries

- Numpy
- Pandas
- MatPLotlib
- DateTime
- SeaBorn
- Mount Google Drive

Function & Method

- hotel_booking_df.info
- hotel_booking_df.drop_duplicates(
- hotel_booking_df.shape[
- hotel_booking_df.isnull() == True
- hotel_booking_df.fillna(np.nan, inplace = True)
- hotel_booking_df.describe()
- hotel_booking_df.drop(['company'], axis=1, inplace=True)
- print(hotel_booking_df.apply(lambda col: col.unique()))
- hotel_booking_df.columns
-

Description of individual Variable

The columns and the data it represents are listed below:

- is_repeated_guest : If the booking was from a repeated guest (1) or not (0)
- previous_cancellations : Number of previous bookings that were cancelled by the customer prior to the current booking
- previous_bookings_not_canceled : Number of previous bookings not cancelled by the customer prior to the current booking
- reserved_room_type : Code of room type reserved
- assigned_room_type : Code of room type assigned
- booking_changes : Number of changes/amendments made to the booking
- deposit_type : Type of the deposit made by the guest
- agent : ID of travel agent who made the booking
- company : ID of the company that made the booking
- days_in_waiting_list : Number of days the booking was in the waiting list
- customer_type : Type of customer, assuming one of four categories
- adr : Average Daily Rate, as defined by dividing the sum of all lodging transactions by the total number of staying nights
- required_car_parking_spaces : Number of car parking spaces required by the customer
- total_of_special_requests : Number of special requests made by the customer
- reservation_status : Reservation status (Canceled, Check-Out or No-Show)

Description of individual Variable

The columns and the data it represents are listed below:

- hotel : Name of the hotel (Resort Hotel or City Hotel)
- is_canceled : If the booking was canceled (1) or not (0)
- lead_time: Number of days before the actual arrival of the guests
- arrival_date_year : Year of arrival date
- arrival_date_month : Month of month arrival date
- arrival_date_week_number : Week number of year for arrival date
- arrival_date_day_of_month : Day of arrival date
- stays_in_weekend_nights : Number of weekend nights (Saturday or Sunday) spent at the hotel by the guests.
- stays_in_week_nights : Number of weeknights (Monday to Friday) spent at the hotel by the guests.
- adults : Number of adults among guests
- children : Number of children among guests
- babies : Number of babies among guests
- meal : Type of meal booked
- country : Country of guests
- market_segment : Designation of market segment
- distribution_channel : Name of booking distribution channel
- reservation_status_date : Date at which the last reservation status was updated

Data Visualisation, Storytelling, and Chart Experimental Design

Function -1

```
# Let's create a function which will give us bar chart of data respective with a col.
def get_count_from_column_bar(df, column_label):
  df_grpd = df[column_label].value_counts()
  df_grpd = pd.DataFrame({'index':df_grpd.index, 'count':df_grpd.values})
  return df_grpd


def plot_bar_chart_from_column(df, column_label, t1):
  df_grpd = get_count_from_column(df, column_label)
  fig, ax = plt.subplots(figsize=(14, 6))
  c= ['g','r','b','c','y']
  ax.bar(df_grpd['index'], df_grpd['count'], width = 0.4, align = 'edge', edgecolor = 'black', linewidth = 4, color = c, linestyle = ':', alpha = 0.5)
  plt.title(t1, bbox={'facecolor':'0.8', 'pad':3})
  plt.legend()
  plt.ylabel('Count')
  plt.xticks(rotation = 15) # use to format the lable of x-axis
  #plt.xlabel(column_label)f
  plt.show()
```

get_count_from_column_bar(df, column_label): This function takes two arguments - a DataFrame (df) and a column label (column_label). It calculates the count of unique values in the specified column using the value_counts() function. Then, it constructs a new DataFrame containing two columns: 'index' (which contains the unique values of the column) and 'count' (which contains the count of occurrences for each unique value). The function returns this DataFrame.

plot_bar_chart_from_column(df, column_label, t1): This function also takes a DataFrame (df), a column label (column_label), and a title (t1). It first calls the get_count_from_column_bar() function to get the DataFrame with the count of unique values for the specified column. Then, it sets up the plot using matplotlib.

It creates a figure and axes using plt.subplots(), specifying the size of the figure.
It defines a list of colors c to be used for the bars in the bar chart.
It uses ax.bar() to plot the bar chart. It uses the 'index' column as the x-values, the 'count' column as the y-values, and applies various formatting parameters (width, edgecolor, linewidth, color, linestyle, alpha) to the bars.
It sets the title of the plot using plt.title() and a bounding box with a light gray background.
It adds a legend (though it seems the legend is not populated with labels, so it might not work as expected).
It sets the y-label using plt.ylabel().
It rotates the x-axis labels by 15 degrees using plt.xticks(rotation=15) for better readability.
It finally displays the plot using plt.show().

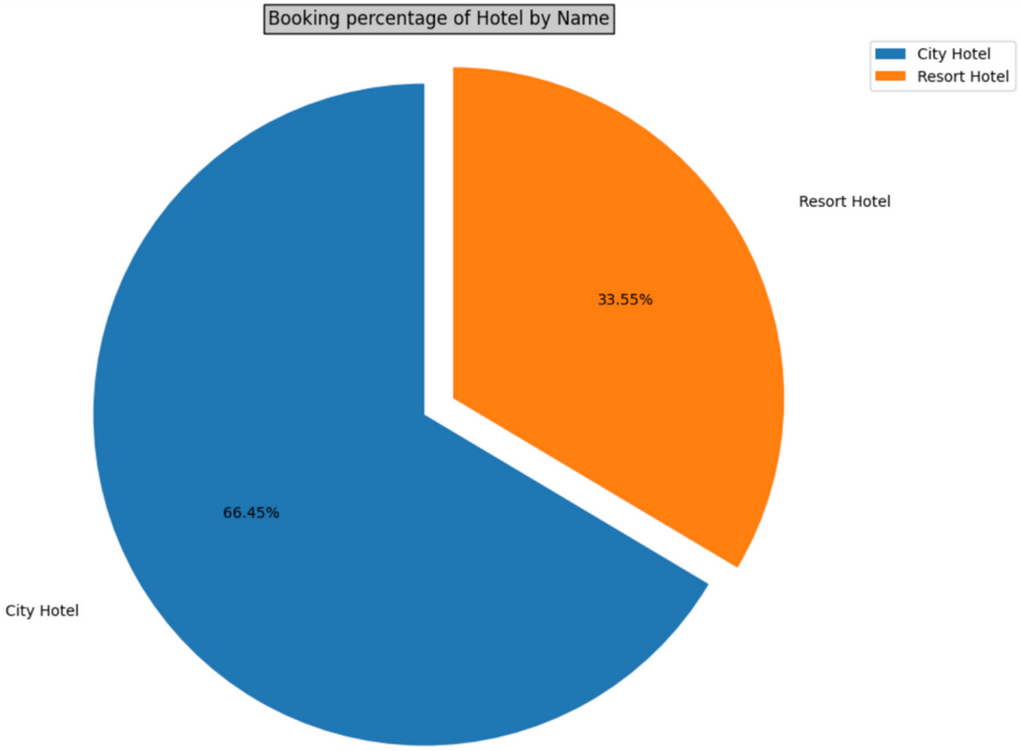
Function -2

```
def get_count_from_column(df, column_label):
 df_grpd = df[column_label].value_counts()
 df_grpd = pd.DataFrame({'index':df_grpd.index,
'count':df_grpd.values})
 return df_grpd

# plot a pie chart from grouped data
def plot_pie_chart_from_column(df, column_label, t1, exp):
 df_grpd = get_count_from_column(df, column_label)
 fig, ax = plt.subplots(figsize=(14,9))
 ax.pie(df_grpd.loc[:, 'count'], labels=df_grpd.loc[:, 'index'],
autopct='%1.2f%%',startangle=90, labeldistance = 1.2, explode = exp)
 plt.title(t1, bbox={'facecolor':'0.8', 'pad':3})
 ax.axis('equal')
 plt.legend()
 plt.show()
```

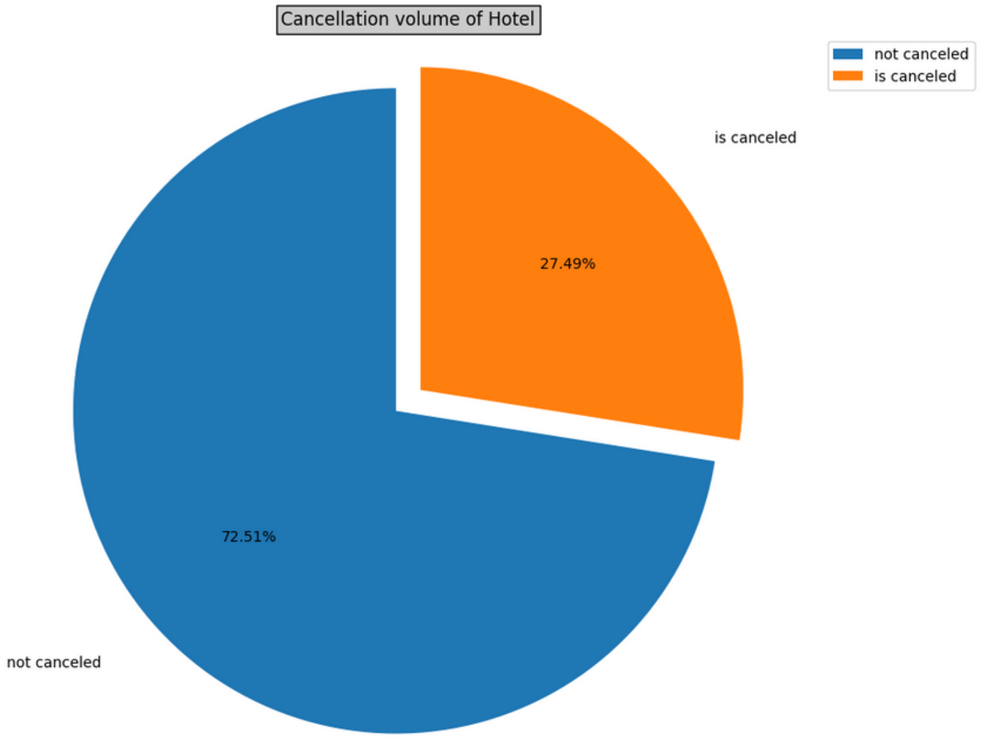
1. get_count_from_column(df, column_label): This function is the same as in your previous code. It takes a DataFrame (df) and a column label (column_label) as arguments. It calculates the count of unique values in the specified column using value_counts(). It constructs a DataFrame with two columns: 'index' (unique values from the column) and 'count' (the count of occurrences for each unique value). The function returns this DataFrame.
2. plot_pie_chart_from_column(df, column_label, t1, exp): This function is used to create and display a pie chart based on the grouped data obtained from the specified column.
    - df_grpd = get_count_from_column(df, column_label): This line calls the previously defined get_count_from_column() function to get the DataFrame with unique values and their counts from the specified column.
    - fig, ax = plt.subplots(figsize=(14,9)): This line initializes a figure and axes for the plot with a specified figsize (width and height).
    - ax.pie(...): This is where the pie chart is created using the grouped data.
        - df_grpd.loc[:, 'count'] extracts the 'count' column from the df_grpd DataFrame.
        - df_grpd.loc[:, 'index'] extracts the 'index' column (unique values) from the df_grpd DataFrame.
        - autopct='%1.2f%%' formats the percentage display on the pie chart.
        - startangle=90 sets the initial angle of the pie chart.
        - labeldistance=1.2 adjusts the distance of labels from the center of the pie.
        - explode=exp is an array of values that determine how much each slice of the pie is offset from the center.
    - plt.title(t1, bbox={'facecolor':'0.8', 'pad':3}): Sets the title of the plot with a light gray bounding box.
    - ax.axis('equal'): Ensures the pie chart is displayed as a circle.
    - plt.legend(): Adds a legend to the pie chart.
    - plt.show(): Finally, displays the pie chart.

Chart-1



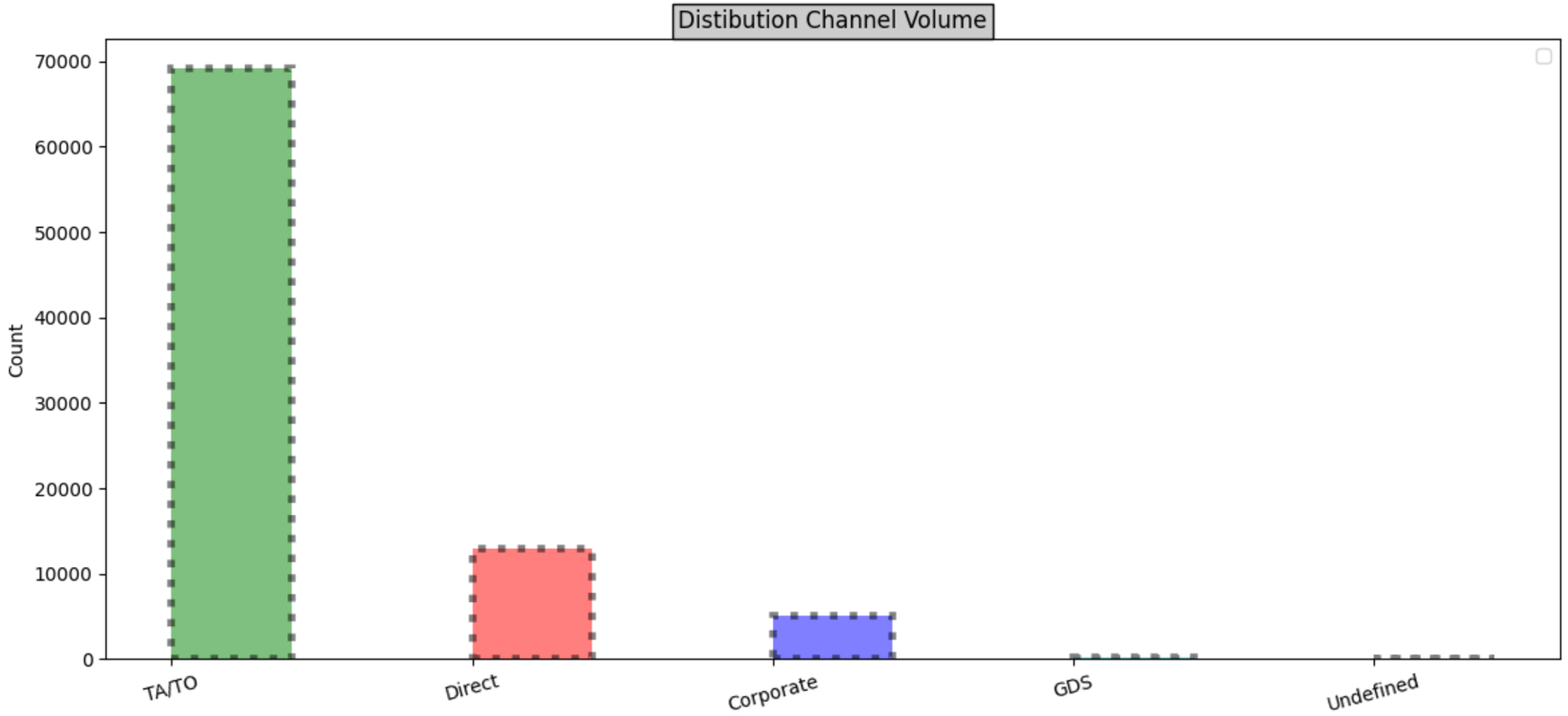
The pie chart shows a significant disparity in hotel bookings between City and Resort hotels, with City Hotel accounting for 61.12% of total bookings and Resort Hotel commanding 38.87%. This disparity highlights the City Hotel's competitive advantage and offers opportunities for both establishments to enhance their business strategies. The City Hotel's comprehensive services and amenities resonate with a large segment of guests, leading to increased revenue and positive business growth. The Resort Hotel can learn from the City Hotel's successful methods and adopt innovative strategies to capture a larger market share. This data-driven insight enables both establishments to leverage their strengths and drive positive business outcomes, fostering growth and excellence in the hospitality industry.

Chart-2



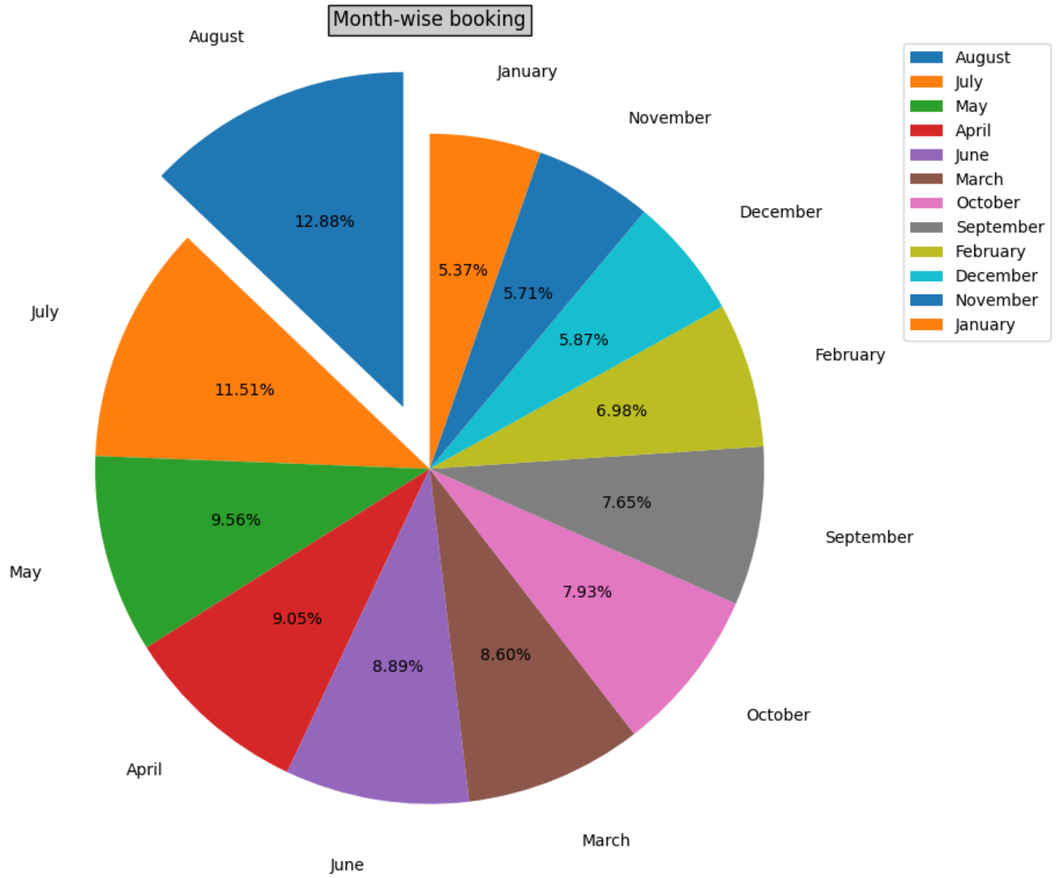
The pie chart shows a significant disparity in hotel bookings between City and Resort hotels, with City Hotel accounting for 61.12% of total bookings and Resort Hotel commanding 38.87%. This disparity highlights the City Hotel's competitive advantage and offers opportunities for both establishments to enhance their business strategies. The City Hotel's comprehensive services and amenities resonate with a large segment of guests, leading to increased revenue and positive business growth. The Resort Hotel can learn from the City Hotel's successful methods and adopt innovative strategies to capture a larger market share. This data-driven insight enables both establishments to leverage their strengths and drive positive business outcomes, fostering growth and excellence in the hospitality industry.

Chart-3



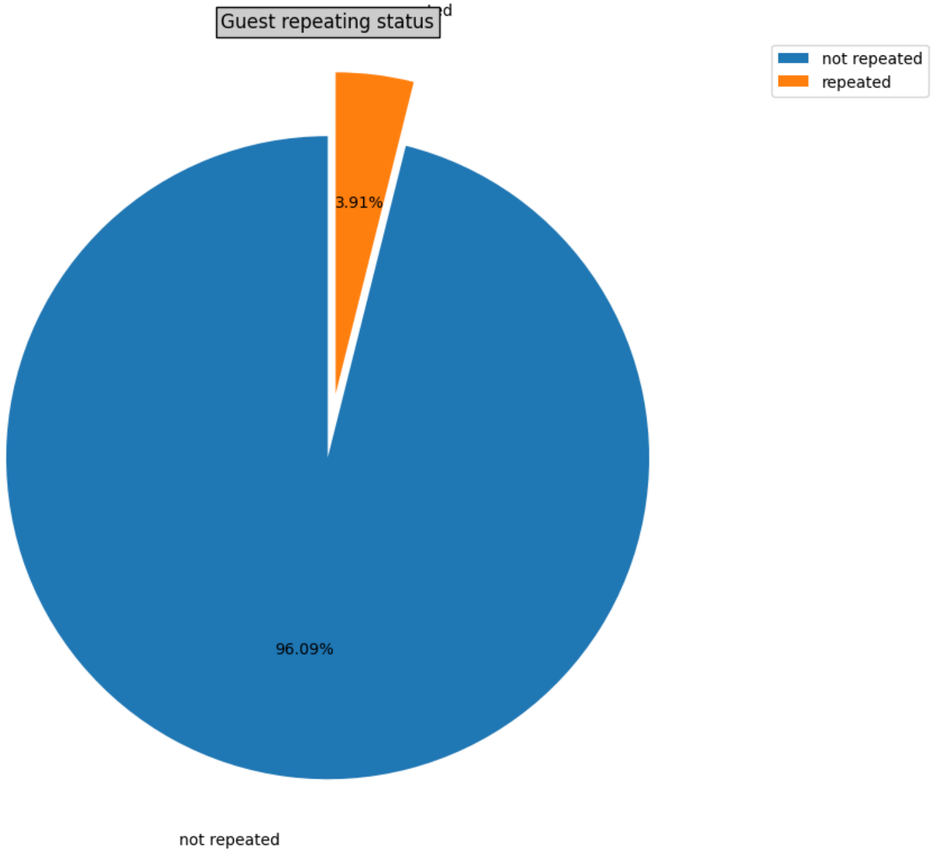
The bar chart displays the maximum volume of bookings across various channels, with the "TA/TO" booking channel dominating. This data highlights its popularity among guests and its role in capturing guest reservations. The strategic recommendation is to continue leveraging and promoting this channel, as it aligns with guest preferences and offers a seamless booking experience. This approach can enhance guest satisfaction and contribute to sustained revenue growth for the hotel. The data-driven insight supports prioritizing and continuing to utilize the "TA/TO" channel for booking purposes, fostering long-term business prosperity and boosting financial performance.

Chart-4



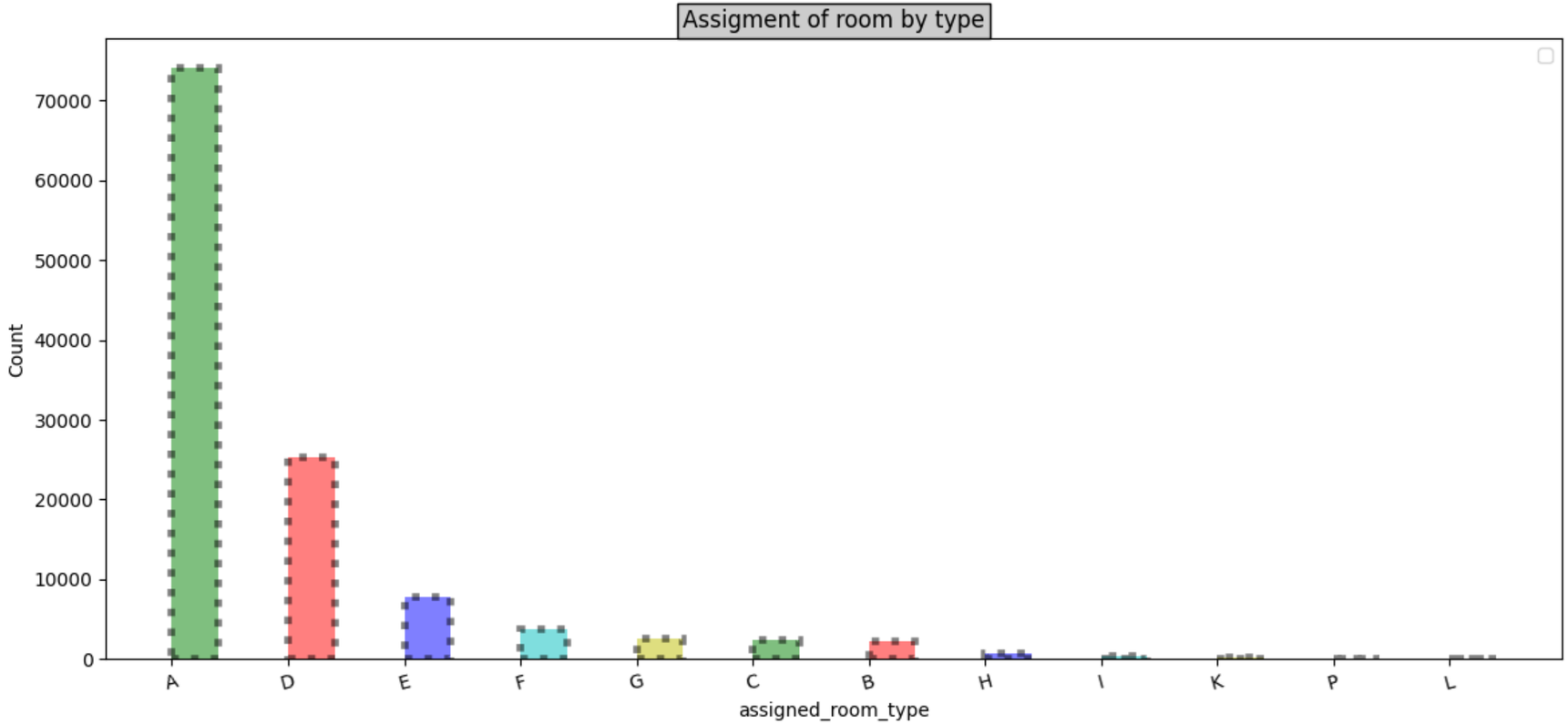
The pie chart shows the distribution of bookings across different months, with May, July, and August having the highest percentage shares. These months are associated with the holiday season, where travelers are more likely to explore accommodations for leisure and vacation purposes. To capitalize on this trend, an aggressive advertisement campaign is suggested to attract a larger customer base during these peak months. This approach can boost occupancy rates and revenue, improve revenue management, and enhance employee satisfaction. Aligning business strategies with seasonal demand can help hotels harness their potential for growth and success in the dynamic hospitality landscape.

Chart-5



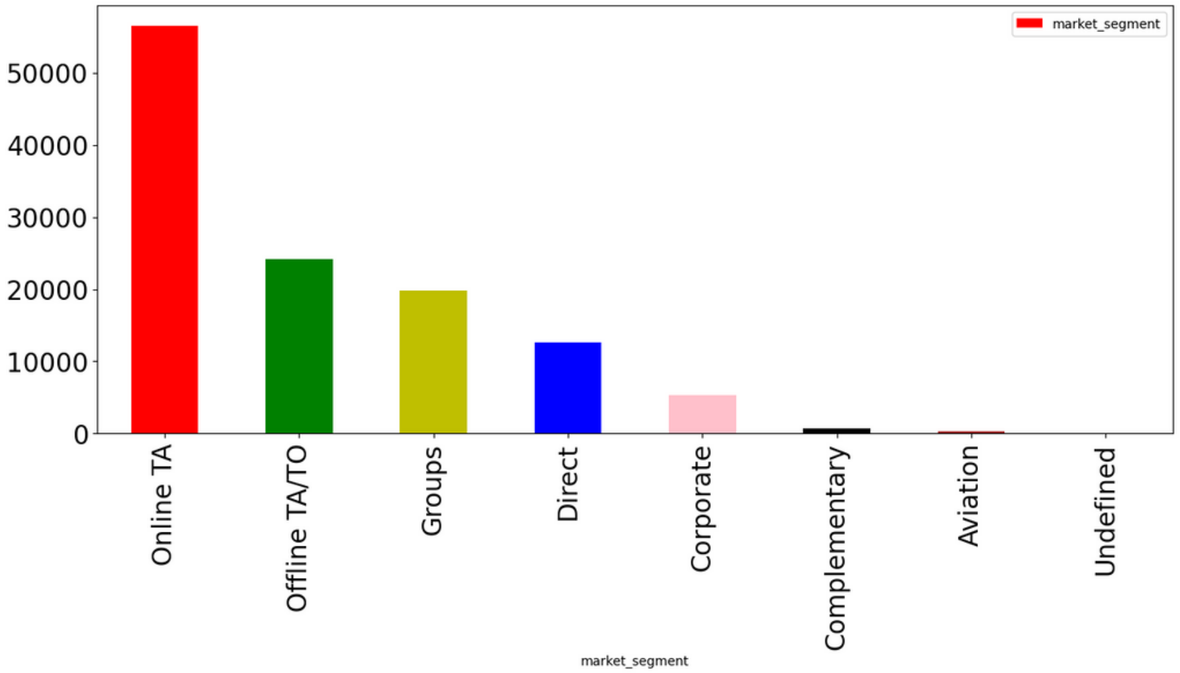
The pie chart shows the distribution of guests into repeated and non-repeated guests. The smaller percentage of repeated guests indicates potential negative growth, possibly due to a lack of effective guest retention strategies. Offering enticing deals to non-repeated customers during off-seasons can convert them into repeat visitors and showcase the hotel's services, amenities, and hospitality. This strategy can enhance revenue, attract new patrons, and foster loyalty and sustained growth.

Chart-6



The pie chart provides a visual representation of room allocation distribution, revealing guest preferences and preferences. "Room Type A" is the frontrunner, with "Room Type D" and "Room Type E" being the most popular options. These preferences are influenced by enhanced services and amenities, which enhance the overall guest experience and contribute to heightened comfort and satisfaction. The alignment between guest preference and room types distinguished by superior services indicates the establishment's effective execution of catering to guest needs and preferences. This preference-driven approach can result in heightened guest satisfaction, favorable reviews, and potential referrals, contributing to an enriched reputation and thriving business trajectory. The pie chart effectively conveys the distribution of allocated room types based on guest preferences, emphasizing the importance of offering enhanced services and fostering a guest-centric ethos that elevates satisfaction, loyalty, and overall success in the competitive hospitality industry.

Chart-7

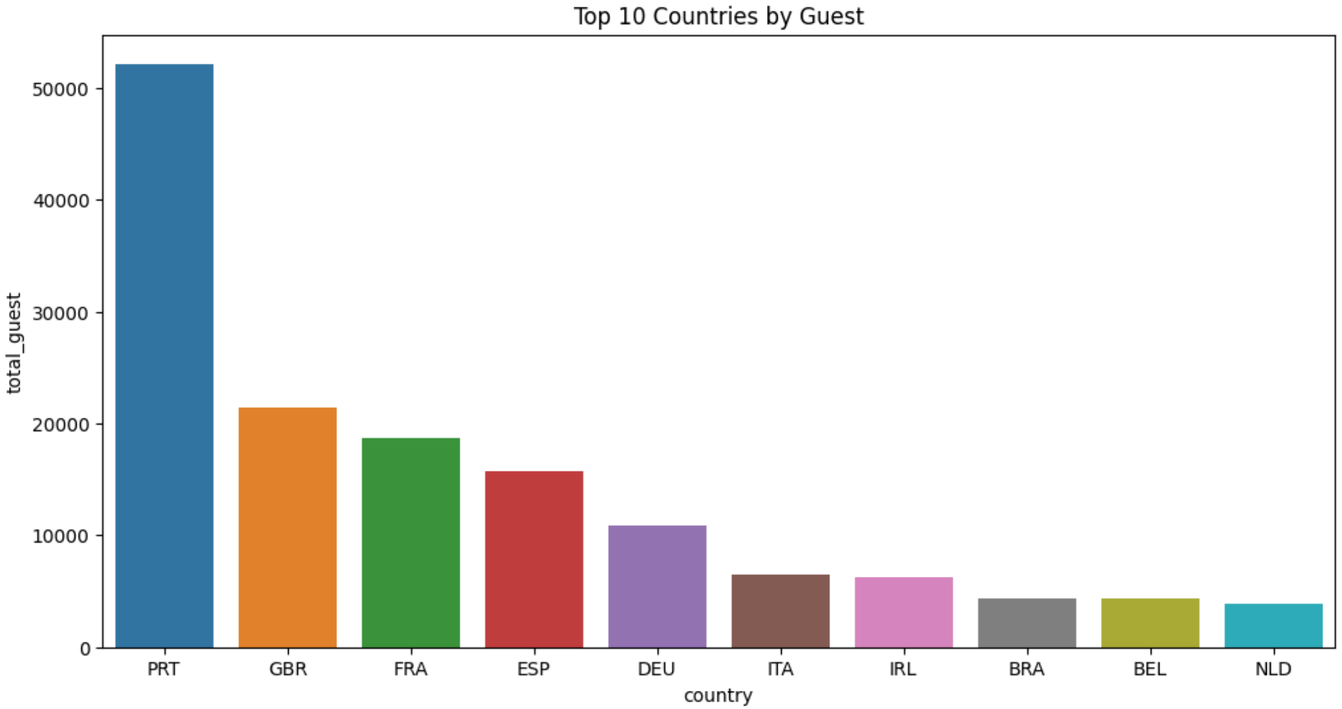


This code creates a new DataFrame called market_segment_df by selecting only the'market_segment' column from the original DataFrame hotel_booking_df. The DataFrame is grouped by the'market_segment' column and the count() function is applied to each group to get the count of occurrences for each unique market segment. The data is sorted in descending order using sort_values(), with the inplace=True argument ensuring sorting is done on the original Series.

A new matplotlib figure is created with a specified width and height, and used as the canvas for the bar plot. A numpy array 'y' is created with three elements: 4, 5, and 6. The bar plot is created using the plot() function from pandas, with the kind='bar' argument specifying a bar plot, color, fontsize, and legend. The show() function is called to display the bar plot on the screen.

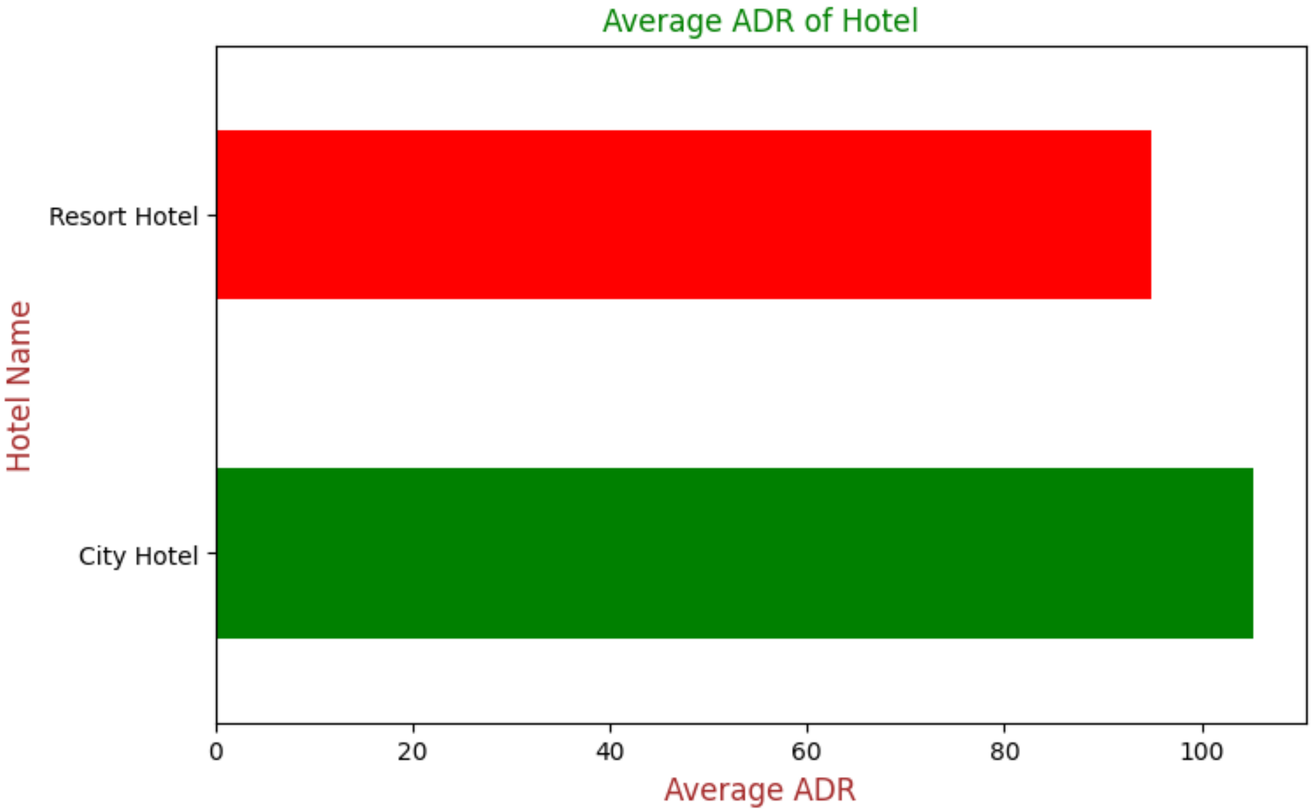
However, there is an issue with using the legend parameter, as it should be True to show the legend, and the array 'y' is not used in the plot.

Chart-8



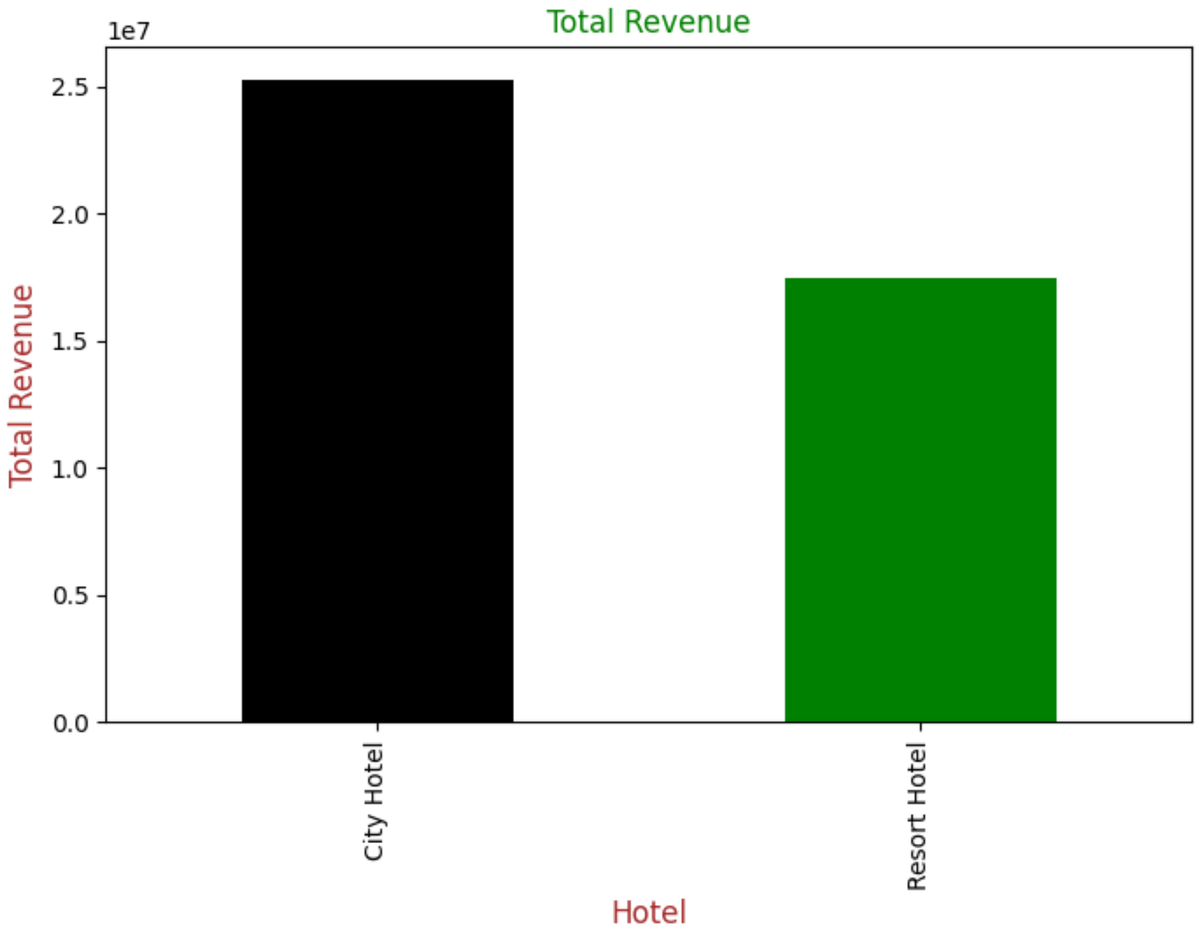
A new DataFrame called guest_country_wise is created by selecting two columns, 'country' and 'total_guest', from the original hotel_booking_df. This DataFrame contains data on each booking's country and the total number of guests associated with that booking. The sum() function is applied to the 'total_guest' column within each group, calculating the total number of guests for each country. The result is a pandas Series with countries as the index and corresponding values representing the total number of guests from each country. The data is sorted in descending order using sort_values(), with the inplace=True argument ensuring sorting is done on the original Series. The first 10 rows are extracted and stored in a new Series called 'top_10_country_by_guest'.

Chart-9



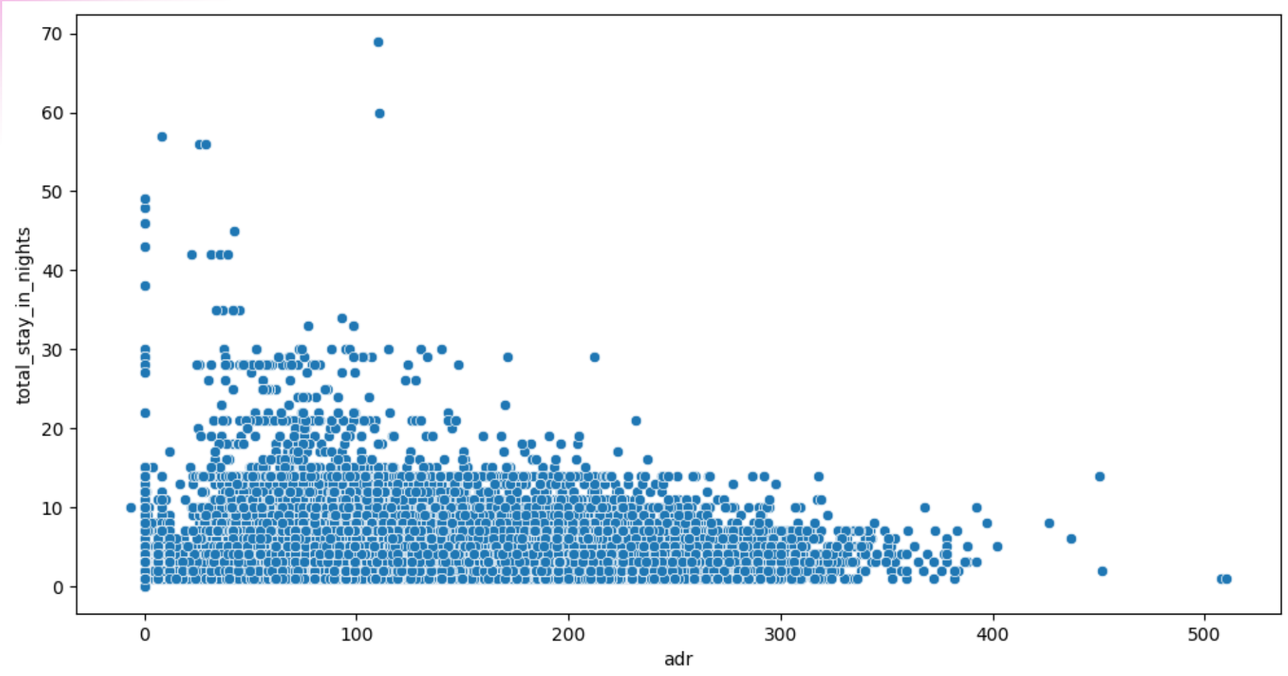
The code calculates the average ADR (Average Daily Rate) of hotels using the DataFrame hotel_booking_df. The mean() function is applied to the 'adr' column for each group, resulting in a pandas Series with hotel names as the index and corresponding values representing the average ADR for each hotel. A matplotlib subplot is created, and the plot() function is used to create a horizontal bar plot with two colors, green ('g') and red ('r'). The x-axis label is set to "Average ADR", and the y-axis label is set to "Hotel Name". The title is set to "Average ADR of Hotel", and the show() function is called to display the bar plot on the screen. The resulting horizontal bar plot displays hotel names on the y-axis and corresponding average ADR values on the x-axis, with green and red bars to differentiate the hotels' average ADRs.

Chart-10



This code calculates the total revenue generated by each hotel in the DataFrame hotel_booking_df and creates a bar plot to visualize and compare the revenue for different hotels. The DataFrame is grouped by the 'hotel' column, and the sum() function is applied to the'revenue' column for each group. The result is a pandas Series with hotel names as the index and corresponding revenue values for each hotel. The bar plot is created using the plot() function from pandas, with the 'kind' argument set to 'bar' and the 'color' argument provided as a tuple of black and green colors. The x-axis label is set to "Hotel", and the y-axis label is set to "Total Revenue". The title is set to "Total Revenue", and the show() function is called to display the bar plot on the screen. The resulting bar plot displays hotel names on the x-axis and corresponding total revenue values on the y-axis, with bars colored black and green to differentiate revenue values for different hotels.

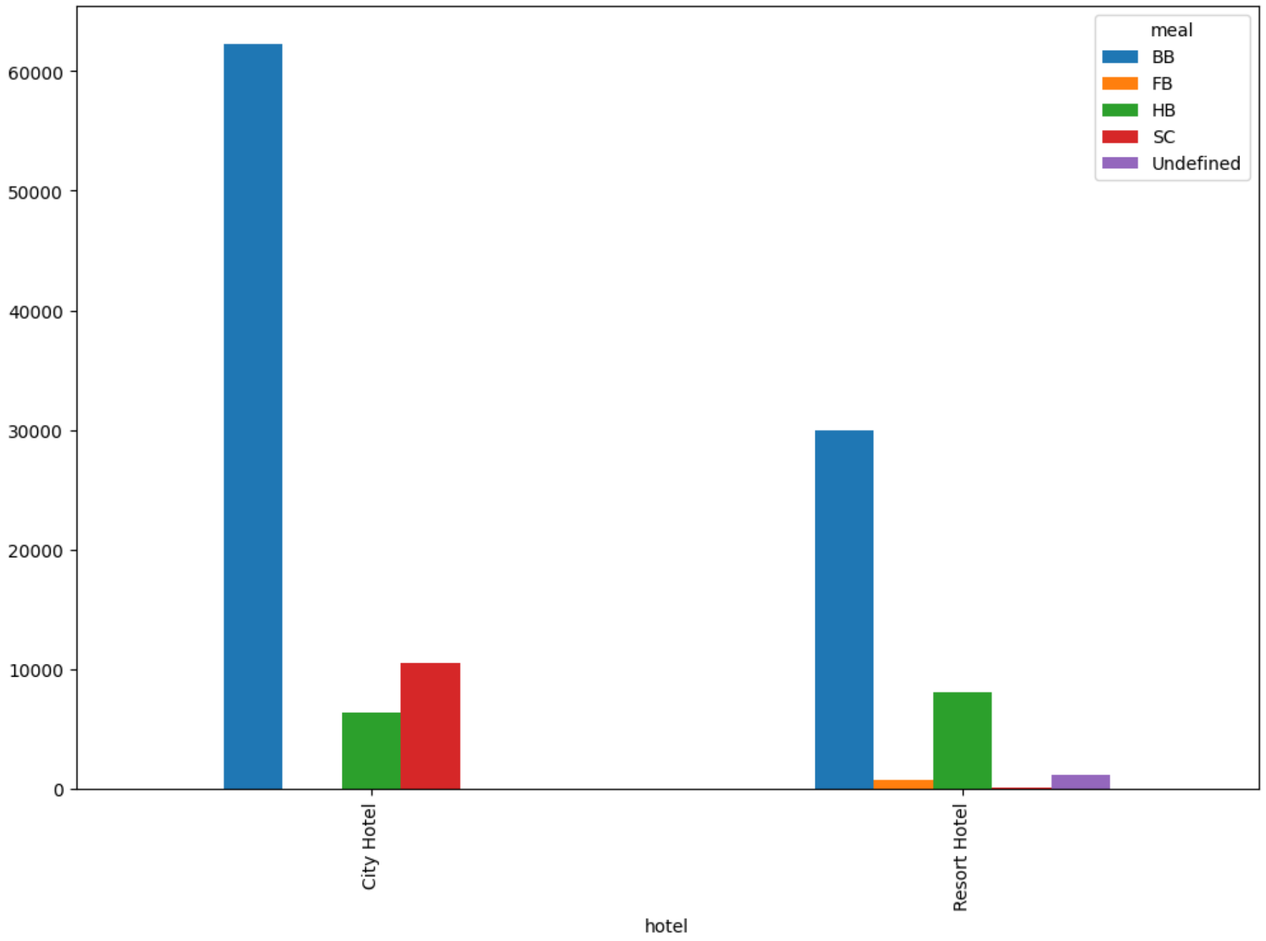
Chart-11



This code creates a scatter plot using Seaborn, a popular Python data visualization library, to visualize the relationship between the average daily rate (adr) and total stay in nights for bookings in the DataFrame hotel_booking_df where the 'adr' value is less than 1000. The plot is created using the'sns.scatterplot()' function, with the 'y' parameter set to 'total_stay_in_nights' and the 'x' parameter set to 'adr'. The data is filtered to include rows with 'adr' values less than 1000.

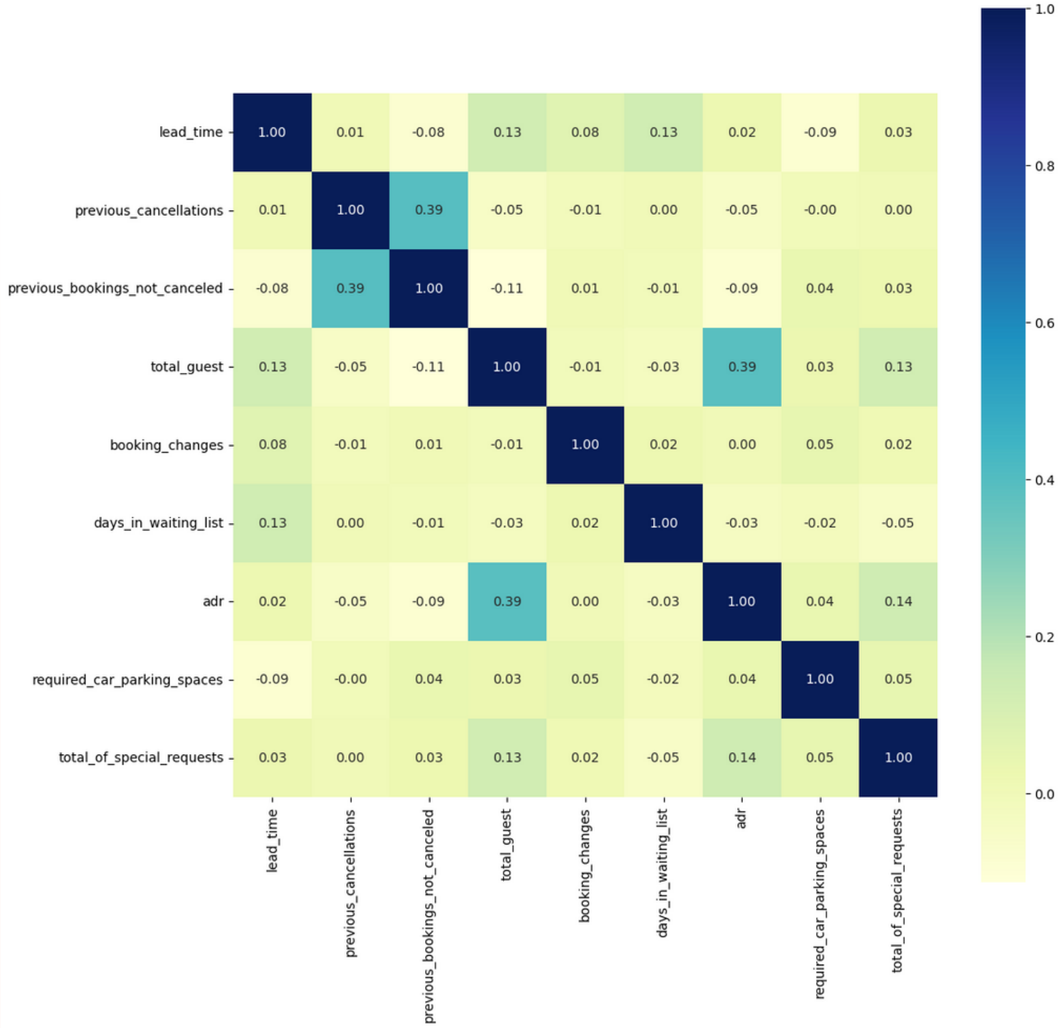
The scatter plot represents the distribution of bookings based on their average daily rate and total number of nights stayed. It can reveal patterns or relationships between the two variables for bookings with an 'adr' value less than 1000.

Chart-12



This code creates a grouped bar chart using pandas and matplotlib to visualize the distribution of meal types for each hotel in the DataFrame hotel_booking_df. The grouped bar chart is created by dividing the DataFrame by two columns, 'hotel' and'meal', and applying the count() function to the'meal' column within each group. The result is a pandas DataFrame with the hotels as the index and the columns representing the count of each meal type for each hotel. The unstack() function pivots the innermost level of the hierarchical index (meal types) to the columns, converting the result into a more suitable format for plotting. The grouped bar chart outputs the data representing the distribution of meal types for each hotel, providing a quick comparison of meal type distributions across different hotels.

Chart-13



This code creates a correlation heatmap using Seaborn to visualize the pairwise correlation between selected numerical columns in the DataFrame hotel_booking_df. The code creates a new DataFrame corr_df by selecting specific numerical columns from the original DataFrame hotel_booking_df. The corr() function calculates the pairwise correlation between these columns, resulting in a DataFrame with correlation coefficients between -1 and 1.

A matplotlib figure and axis are created, and the heatmap is created using Seaborn's heatmap() function. The data parameter is set to corr_df, and the annotations are formatted to display two decimal places. The heatmap is then colored Yellow-Green-Blue, indicating the color gradient for the heatmap. The heatmap displays the pairwise correlation coefficients between the selected numerical columns, with positive correlations represented by lighter colors and negative correlations by darker colors. This visualization helps identify relationships between variables and detect patterns in the data.

The answer to the business objective

The following business aim was met :

1. In order for the hotel industry to prosper, a few factors must be taken into account, including high revenue generation, customer happiness, and employee retention.
2. By using a pie chart distribution, we can demonstrate to the client which months generate the most money.
3. Increasing revenue by using a bar chart to show which types of rooms are most frequently booked and when visitors are most likely to travel.
4. As a result, the customer can be properly prepared in advance, minimising long-term complaints and contributing to further improvement of their hospitality.
5. To encourage clients to contact offices for bulk reservations during the off-season, outliers such as larger visitor numbers than average were sprinkled across the plot. This helped generate more money.
6. We can display the visitor arrivals trend at client venues, allowing clients to schedule visitors in advance for their entertainment and leisure activities.
7. In order for the percentages underneath those numbers to be improved by a variety of mediums, we were also able to correlate the values indicating the maximum and minimum % between them.

Conclusion

1. City Hotel generates greater income and profit and appears to be more popular among travellers.
2. Compared to the other months, the majority of reservations are made in July and August.
3. Travellers favour accommodation Type A over all other accommodation types.
4. Portugal and the United Kingdom make the most reservations.
5. The majority of visitors stay in hotels for 1-4 days.
6. The City Hotel keeps a larger number of visitors.
7. About one-fourth of all reservations are cancelled. City Hotel is the source of more cancellations.
8. Compared to returning consumers, new guests frequently cancel reservations.
9. Booking cancellations are unaffected by lead time, waiting list length, or client assignment of a reserved room.
10. Corporate has the most percentage of repeated guests while TA/TO has the least whereas in the case of cancelled bookings TA/TO has the most percentage while Corporate has the least.
11. The length of the stay decreases as ADR increases probably to reduce the cost.

Hurrah! You achieved your EDA Capstone Project successfully!