**INTRODUCTION**

* 1. **About the Project**

In data science, a hotel booking system leverages advanced analytics and machine learning to optimize operations and enhance customer experience. By analyzing historical booking data, the system can forecast demand, enabling dynamic pricing strategies that adjust rates in real-time based on factors like occupancy levels and market trends. It also employs customer segmentation to tailor marketing efforts and recommendation engines to suggest personalized offers, improving guest satisfaction. Additionally, data-driven insights streamline operational efficiency and sentiment analysis of guest reviews helps refine service quality, ultimately driving better business outcomes.

* 1. **Problem Statement**

Problem title:

“HOTEL BOOKING ANALYSIS”

Problem overview:

This project aims toanalyze hotel booking data to address challenges like high cancellation rates, fluctuating demand, and inefficient resource allocation. By examining booking trends, predicting cancellations, optimizing pricing strategies, and enhancing customer segmentation, we seek to improve operational efficiency, boost revenue, and enhance the overall guest experience.

**1.3 Objectives of the project**

The objectives of this project are to:

1. Identify Booking Trends: Analyze seasonal patterns and customer demographics to understand booking behaviors.
2. Predict Cancellations: Develop predictive models to anticipate booking cancellations and improve demand management.
3. Optimize Pricing Strategies: Evaluate and refine pricing strategies to maximize revenue based on demand forecasts.
4. Enhance Customer Segmentation: Segment customers to tailor marketing efforts and service offerings for improved satisfaction**.**

**SYSTEM REQUIREMENTS AND SPECIFICATIONS**

1. Operating System:

- Windows 10 or later

- macOS 10.14 or later

- Linux (any modern distribution)

2. Python Version:

- Python 3.6 or later (recommended Python 3.8 or newer for compatibility with the latest libraries)

3. Memory (RAM):

- Minimum: 4 GB

- Recommended: 8 GB or more, depending on the size of the dataset

4. Storage:

- Minimum: 1 GB of free disk space for installing Python and libraries

- Additional space may be required for data files and results

5. Processor:

- Minimum: Intel Core i3 or equivalent

- Recommended: Intel Core i5 or higher, or AMD equivalent for faster processing

**2.1 SOFTWARE REQUIREMENTS**

1 Python: The core programming language for the project.

2 IDE/Code Editor:

* Jupyter Notebook or JupyterLab: For interactive coding and visualization.
* PyCharm or VS Code: For a more feature-rich development environment.

3 Data Processing and Analysis Libraries:

* Pandas: For data manipulation and analysis.
* NumPy: For numerical operations and array handling.

4 Data Visualization Libraries:

* Matplotlib: For basic plotting and data visualization.
* Seaborn: For statistical data visualization and attractive plots.
* Plotly: For interactive and web-based visualizations.

5 Machine Learning Libraries:

* Scikit-learn: For building and evaluating machine learning models.
* Statsmodels: For statistical modeling and hypothesis testing.

6 Data Cleaning and Preprocessing Libraries:

* Openpyxl or xlrd/xlwt: For handling Excel files if needed.
* BeautifulSoup or Requests: For web scraping, if necessary.

7 Dataset Sources:

* Kaggle: Datasets such as:
  + Hotel Booking Demand Dataset (Link)
  + Hotel Booking Dataset (Link)

8 Environment Management:

* Anaconda: For managing Python environments and packages.
* Virtualenv: For creating isolated Python environments.

**2.2 HARDWARE REQUIREMENTS**

1 **Processor (CPU):**

* **Minimum:** Intel i5 or equivalent
* **Recommended:** Intel i7 or AMD Ryzen 7 or better
* **Reason:** A more powerful processor will handle data processing and computations more efficiently, especially for large datasets or complex models.

2 **Memory (RAM):**

* **Minimum:** 8 GB
* **Recommended:** 16 GB or more
* **Reason:** Sufficient RAM is crucial for handling large datasets and running multiple processes simultaneously.

3 S**torage:**

* **Minimum:** 256 GB SSD
* **Recommended:** 512 GB SSD or higher
* **Reason:** An SSD provides faster data access and processing speeds. Adequate storage is necessary to save datasets, code, and results.

4 **Graphics Processing Unit (GPU):**

* **Minimum:** Integrated graphics (e.g., Intel HD Graphics)
* **Recommended:** Dedicated GPU (e.g., NVIDIA GeForce GTX 1660 or better)
* **Reason:** A dedicated GPU can accelerate machine learning and data processing tasks, particularly for deep learning models.

5 **Network:**

* **Minimum:** Stable internet connection
* **Recommended:** High-speed internet
* **Reason:** A stable and fast internet connection is essential for downloading datasets from Kaggle and performing cloud-based tasks.

6 **Other Considerations:**

* **External Backup:** An external hard drive or cloud storage for data backup and additional storage.
* **Peripherals:** A high-resolution monitor, keyboard, and mouse to enhance productivity.

**IMPLEMENTATION**

**3.1 Function method description**

**1.** **Importing Libraries**

* import pandas as pd: Imports the pandas library as pd, which is used for data manipulation and analysis.
* import matplotlib.pyplot as plt: Imports the pyplot module from matplotlib as plt, which is used for creating plots and visualizations.
* import seaborn as sns: Imports the seaborn library as sns, which is used for making statistical plots with additional aesthetics.
* import warnings: Imports the warnings library to handle warnings that may arise during code execution (though not used explicitly here).

1. **Loading Data**

* pd.read\_csv('hotel\_bookings.csv', delimiter='\t'): Reads the CSV file named 'hotel\_bookings.csv' using a tab (\t) as the delimiter, and loads it into a DataFrame called df.

**3 Exploring Data**

* **df.head()**: Shows the first 5 rows of the DataFrame. Useful for getting a quick overview of the data.
* **df.head(10)**: Shows the first 10 rows of the DataFrame. Adjust the number to view more rows.
* **df.tail()**: Shows the last 5 rows of the DataFrame. Useful for viewing the end of the dataset.
* **df.tail(8)**: Shows the last 8 rows of the DataFrame.
* **df.shape**: Returns a tuple (number of rows, number of columns) indicating the size of the DataFrame.
* **df.columns**: Lists the column names in the DataFrame.
* **df.info()**: Provides a summary including the number of non-null entries and data types for each column.
* **df.describe(include='object')**: Summarizes statistics for categorical (object) columns, such as counts and unique values.

**4 Handling Missing Values**

* **df.isnull()**: Returns a DataFrame of boolean values (True for missing values, False otherwise).
* **df.isnull().sum()**: Sums the boolean values to get the count of missing values in each column.
* **df.drop(['company', 'agent'], axis=1, inplace=True)**: Drops the columns company and agent from the DataFrame (axis=1 specifies columns). inplace=True modifies the DataFrame in place without creating a copy.
* **df.dropna(inplace=True)**: Removes any rows with missing values. inplace=True ensures the DataFrame is updated directly.
* **df.isnull().sum()**: Re-checks for any remaining missing values after the cleanup.

**5 Visualizing Data**

* df['adr'].plot(kind='box'): Creates a boxplot of the adr (Average Daily Rate) column to visualize the distribution and identify outliers.
* df['is\_canceled'].value\_counts(normalize=True): Calculates the proportion of each unique value in the is\_canceled column. normalize=True returns proportions rather than counts.
* print(cancelled\_prec): Outputs the proportion of canceled versus non-canceled reservations.
* **plt.figure(figsize=(5, 4))**: Creates a new figure for plotting with specified dimensions (5 inches by 4 inches).
* **plt.title('Reservation\_Status\_Count')**: Sets the title of the plot.
* **plt.bar(['Not\_canceled', 'canceled'], df['is\_canceled'].value\_counts(), edgecolor='k', width=0.7)**: Creates a bar chart showing the count of canceled and not canceled reservations. edgecolor='k' adds a black border around the bars. width=0.7 sets the width of the bars.
* **plt.show()**: Displays the plot.
* **df['is\_canceled'].replace({0: 'not canceled', 1: 'canceled'})**: Replaces numeric values (0 and 1) in the is\_canceled column with descriptive labels ('not canceled' and 'canceled').
* **plt.figure(figsize=(8, 4))**: Creates a new figure with specified dimensions (8 inches by 4 inches).
* **sns.countplot(x='hotel', hue='is\_canceled', data=df, palette='Blues')**: Creates a count plot showing the number of reservations for each hotel type, with different colors for canceled and non-canceled reservations.
* **ax1.legend(bbox\_to\_anchor=(1, 1))**: Positions the legend outside the plot area.
* **legend\_labels, \_ = ax1.get\_legend\_handles\_labels()**: Retrieves legend handles and labels for further customization.
* **plt.title('Reservation Status in different hotels', size=20)**: Sets the title of the plot with a font size of 20.
* **plt.show()**: Displays the plot.
* **resort\_hotel = df[df['hotel'] == 'Resort Hotel']**: Filters the DataFrame to include only rows where the hotel column is 'Resort Hotel'.
* **City\_hotel = df[df['hotel'] == 'City Hotel']**: Filters the DataFrame to include only rows where the hotel column is 'City Hotel'.
* **plt.figure(figsize=(8, 6))**: Creates a new figure with dimensions (8 inches by 6 inches).
* **plt.title('Average Daily Rate in City and Resort Hotel', fontsize=30)**: Sets the title of the plot with a font size of 30.
* **plt.plot(resort\_hotel.index, resort\_hotel['adr'], label='Resort Hotel')**: Plots the adr values for Resort Hotel over time. Uses the DataFrame index for the x-axis.
* **plt.plot(City\_hotel.index, City\_hotel['adr'], label='City Hotel')**: Plots the adr values for City Hotel over time.
* **plt.legend()**: Adds a legend to differentiate between Resort and City hotels.
* **plt.show()**: Displays the plot
* **df.groupby('month')['adr'].mean()**: Groups the data by the month column and calculates the mean adr for each month.
* **.reset\_index()**: Resets the index of the resulting DataFrame to make month a column rather than an index.
* plt.figure(figsize=(10, 6)): Creates a new figure with dimensions (10 inches by 6 inches).
* sns.barplot(x='month', y='adr', data=average\_daily\_rate, color='skyblue'): Creates a bar plot showing the average daily rate (adr) for each month.
* plt.xlabel('Month'): Labels the x-axis as 'Month'.
* plt.ylabel('Average Daily Rate'): Labels the y-axis as 'Average Daily Rate'.
* plt.title('Average Daily Rate per Month'): Sets the title of the plot.
* **Cancelled\_data = df[df['is\_canceled'] == 1]**: Filters the DataFrame to include only rows where is\_canceled is 1 (canceled reservations).
* **top\_10\_Country = Cancelled\_data['country'].value\_counts()[:10]**: Counts the number of canceled reservations by country and selects the top 10 countries.
* **plt.figure(figsize=(8, 8))**: Creates a new figure with dimensions (8 inches by 8 inches).
* **plt.title('Top 10 Countries with reservation cancelled')**: Sets the title of the plot.
* **plt.pie(top\_10\_Country, autopct='%.2f', labels=top\_10\_Country.index)**: Creates a pie chart with the top 10 countries. autopct='%.2f' formats the percentage labels to two decimal places
* **df.reset\_index()**: Resets the index of the DataFrame, which is useful for grouping operations.
* **.groupby("hotel").aggregate({"hotel": "count"})**: Groups the DataFrame by the hotel column and counts the number of entries in each group.
* **rename(columns={'hotel': 'count', 'index': 'hotel'})**: Renames columns for clarity; hotel to count and index to hotel.
* **sort\_values('count', ascending=False)**: Sorts the DataFrame by the count column in descending order.
* **.reset\_index()**: Resets the index of the sorted DataFrame..
* **plt.figure(figsize=(8, 8))**: Creates a new figure with dimensions (8 inches by 8 inches).
* **explode = [0.07, 0]**: Defines the fraction of the radius to offset each slice. The first slice is slightly exploded.
* **labels = hotel["hotel"]**: Sets the labels for each slice of the pie chart.
* **colors = ['blue', 'Red']**: Defines the colors for each slice.
* **plt.pie(hotel["count"], autopct='%.1f%%', explode=explode, labels=labels, colors=colors)**: Creates a pie chart with the defined settings.
* **plt.title(label="Type of Hotel", fontsize=16)**: Sets the title of the pie chart with a font size of 16.
* **plt.figure(figsize=(13, 6))**: Creates a new figure with dimensions (13 inches by 6 inches).
* **data4 = df[df["is\_canceled"] == 1].reset\_index()**: Filters the DataFrame to include only canceled reservations and resets the index.
* **cancellations = data4[["hotel", "is\_canceled", "arrival\_date\_year", "arrival\_date\_month"]]**: Selects relevant columns for analysis.
* **ordered\_months = [...]**: Defines a list of months in a specific order for plotting.
* **cancellations["arrival\_date\_month"] = pd.Categorical(cancellations["arrival\_date\_month"], categories=ordered\_months, ordered=True)**: Converts the arrival\_date\_month column to a categorical type with an ordered list of months.
* **sns.countplot(x="arrival\_date\_month", hue="hotel", data=cancellations)**: Creates a count plot of cancellations by month, with bars colored by hotel type.
* **plt.title(label="Cancellations Each Month", fontsize=15)**: Sets the title of the plot with a font size of 15.
* **plt.legend(loc="upper right")**: Positions the legend in the upper right corner of the plot.
* **plt.tight\_layout()**: Adjusts the plot layout to fit all elements.
* **plt.show()**: Displays the plot.

**PSEUDO CODE**

# Import necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

# Suppress warnings

warnings.filterwarnings('ignore')

# Load dataset

df = pd.read\_csv('hotel\_bookings.csv', delimiter='\t')

# Explore data

print("First 5 rows:")

print(df.head())

print("\nFirst 10 rows:")

print(df.head(10))

print("\nLast 5 rows:")

print(df.tail())

print("\nLast 8 rows:")

print(df.tail(8))

print("\nData shape (rows, columns):", df.shape)

print("\nColumn names:", df.columns)

print("\nData information:")

print(df.info())

print("\nSummary statistics for categorical columns:")

print(df.describe(include='object'))

# Handle missing values

print("\nMissing values in each column:")

print(df.isnull().sum())

# Drop unnecessary columns and rows with missing values

df.drop(['company', 'agent'], axis=1, inplace=True)

df.dropna(inplace=True)

print("\nMissing values after cleanup:")

print(df.isnull().sum())

# Visualize data

# Boxplot for 'adr'

plt.figure(figsize=(8, 6))

plt.title('Boxplot of Average Daily Rate (ADR)')

df['adr'].plot(kind='box')

plt.show()

# Proportion of cancellations

cancelled\_prec = df['is\_canceled'].value\_counts(normalize=True)

print("\nProportion of cancellations:")

print(cancelled\_prec)

# Bar chart for reservation status

plt.figure(figsize=(5, 4))

plt.title('Reservation Status Count')

plt.bar(['Not Canceled', 'Canceled'], df['is\_canceled'].value\_counts(), edgecolor='k', width=0.7)

plt.show()

# Replace numeric values in 'is\_canceled' with descriptive labels

df['is\_canceled'] = df['is\_canceled'].replace({0: 'not canceled', 1: 'canceled'})

# Count plot of reservation status by hotel type

plt.figure(figsize=(8, 4))

ax1 = sns.countplot(x='hotel', hue='is\_canceled', data=df, palette='Blues')

ax1.legend(bbox\_to\_anchor=(1, 1))

plt.title('Reservation Status in Different Hotels', size=20)

plt.show()

# Filter data for Resort and City Hotels

resort\_hotel = df[df['hotel'] == 'Resort Hotel']

city\_hotel = df[df['hotel'] == 'City Hotel']

# Plot ADR over time for Resort and City Hotels

plt.figure(figsize=(8, 6))

plt.title('Average Daily Rate in City and Resort Hotels', fontsize=30)

plt.plot(resort\_hotel.index, resort\_hotel['adr'], label='Resort Hotel')

plt.plot(city\_hotel.index, city\_hotel['adr'], label='City Hotel')

plt.legend()

plt.show()

# Calculate and plot average daily rate per month

average\_daily\_rate = df.groupby('month')['adr'].mean().reset\_index()

plt.figure(figsize=(10, 6))

sns.barplot(x='month', y='adr', data=average\_daily\_rate, color='skyblue')

plt.xlabel('Month')

plt.ylabel('Average Daily Rate')

plt.title('Average Daily Rate per Month')

plt.show()

# Top 10 countries with the highest number of canceled reservations

cancelled\_data = df[df['is\_canceled'] == 'canceled']

top\_10\_countries = cancelled\_data['country'].value\_counts()[:10]

plt.figure(figsize=(8, 8))

plt.title('Top 10 Countries with Reservations Cancelled')

plt.pie(top\_10\_countries, autopct='%.2f%%', labels=top\_10\_countries.index)

plt.show()

# Hotel proportion pie chart

hotel\_counts = df.reset\_index().groupby("hotel").size().reset\_index(name='count')

hotel\_counts = hotel\_counts.sort\_values('count', ascending=False)

plt.figure(figsize=(8, 8))

explode = [0.07, 0]

labels = hotel\_counts["hotel"]

colors = ['blue', 'red']

plt.pie(hotel\_counts["count"], autopct='%.1f%%', explode=explode, labels=labels, colors=colors)

plt.title("Proportion of Hotel Types", fontsize=16)

plt.show()

# Cancellations each month, categorized by hotel type

plt.figure(figsize=(13, 6))

cancellations = df[df["is\_canceled"] == 'canceled'][["hotel", "is\_canceled", "arrival\_date\_year", "arrival\_date\_month"]]

ordered\_months = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]

cancellations["arrival\_date\_month"] = pd.Categorical(cancellations["arrival\_date\_month"], categories=ordered\_months, ordered=True)

sns.countplot(x="arrival\_date\_month", hue="hotel", data=cancellations)

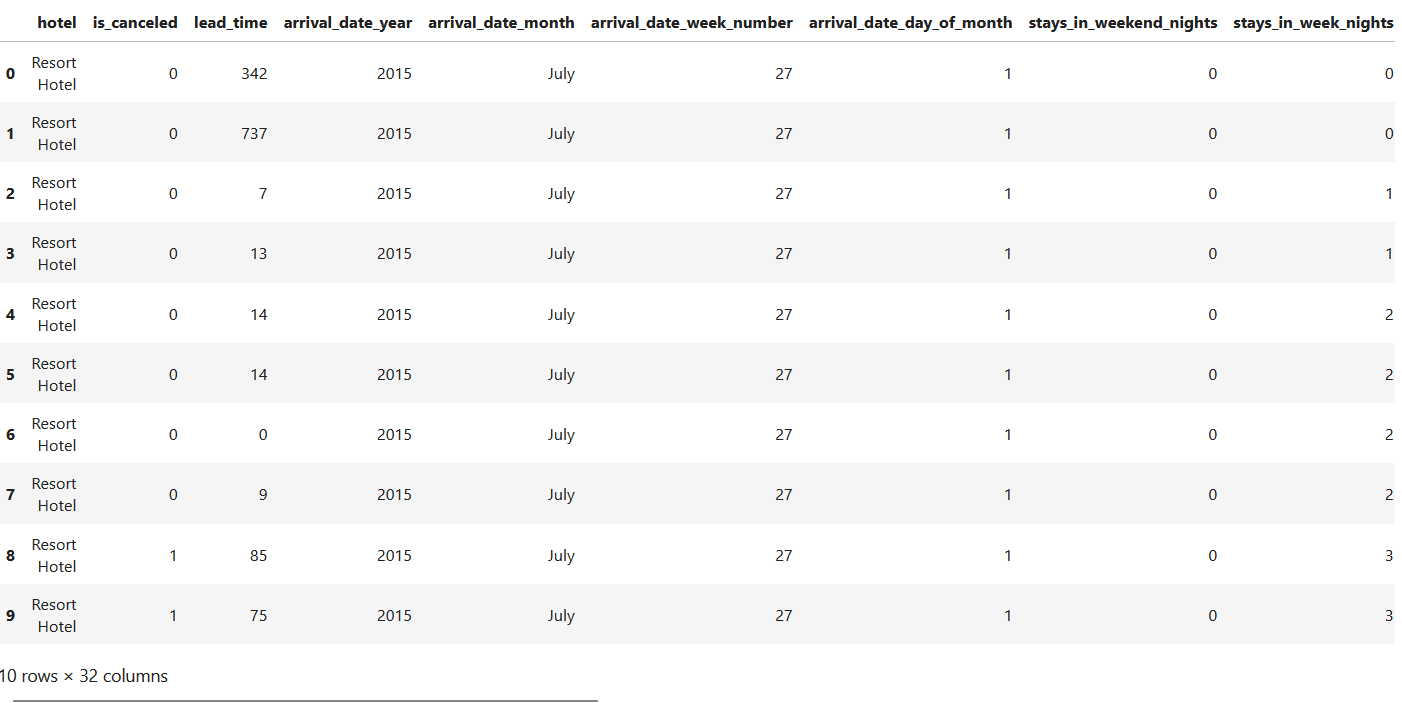
plt.title("Cancellations Each Month", fontsize=15)

plt.legend(loc="upper right")

plt.tight\_layout()

plt.show()

**3.2 RESULTS (Screen shorts of the outputs)**





(119390, 32)

Index(['hotel','is\_canceled','lead\_time','arrival\_date\_year','arrival\_date\_month','arrival\_date\_week\_number','arrival\_date\_day\_of\_month','stays\_in\_weekend\_nights’,'stays\_in\_week\_nights', 'adults','children','babies','meal',’'country’,'market\_segment','distribution\_channel','is\_repeated\_guest','previous\_cancellations','previous\_bookings\_not\_canceled', 'reserved\_room\_type','assigned\_room\_type','booking\_changes','deposit\_type','agent', 'company','days\_in\_waiting\_list','customer\_type','adr','required\_car\_parking\_spaces', 'total\_of\_special\_requests','reservation\_status','reservation\_status\_date'],dtype='object')

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 119390 entries, 0 to 119389

Data columns (total 32 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 hotel 119390 non-null object

1 is\_canceled 119390 non-null int64

2 lead\_time 119390 non-null int64

3 arrival\_date\_year 119390 non-null int64

4 arrival\_date\_month 119390 non-null object

5 arrival\_date\_week\_number 119390 non-null int64

6 arrival\_date\_day\_of\_month 119390 non-null int64

7 stays\_in\_weekend\_nights 119390 non-null int64

8 stays\_in\_week\_nights 119390 non-null int64

9 adults 119390 non-null int64

10 children 119386 non-null float64

11 babies 119390 non-null int64

12 meal 119390 non-null object

13 country 118902 non-null object

14 market\_segment 119390 non-null object

15 distribution\_channel 119390 non-null object

16 is\_repeated\_guest 119390 non-null int64

17 previous\_cancellations 119390 non-null int64

18 previous\_bookings\_not\_canceled 119390 non-null int64

19 reserved\_room\_type 119390 non-null object

20 assigned\_room\_type 119390 non-null object

21 booking\_changes 119390 non-null int64

22 deposit\_type 119390 non-null object

23 agent 103050 non-null float64

24 company 6797 non-null float64

25 days\_in\_waiting\_list 119390 non-null int64

26 customer\_type 119390 non-null object

27 adr 119390 non-null float64

28 required\_car\_parking\_spaces 119390 non-null int64

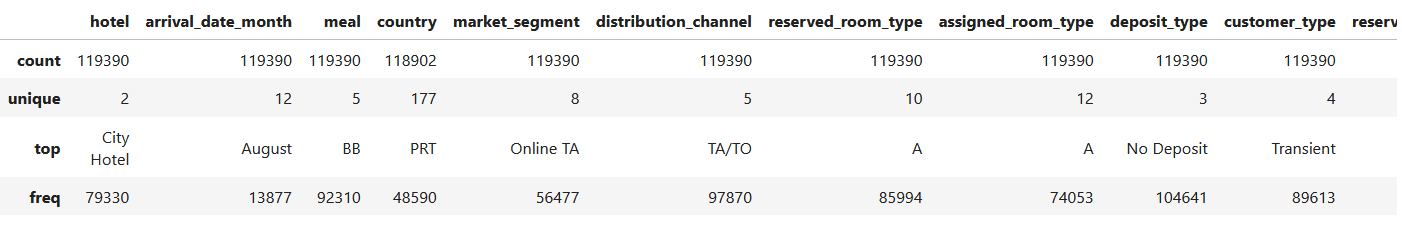
29 total\_of\_special\_requests 119390 non-null int64

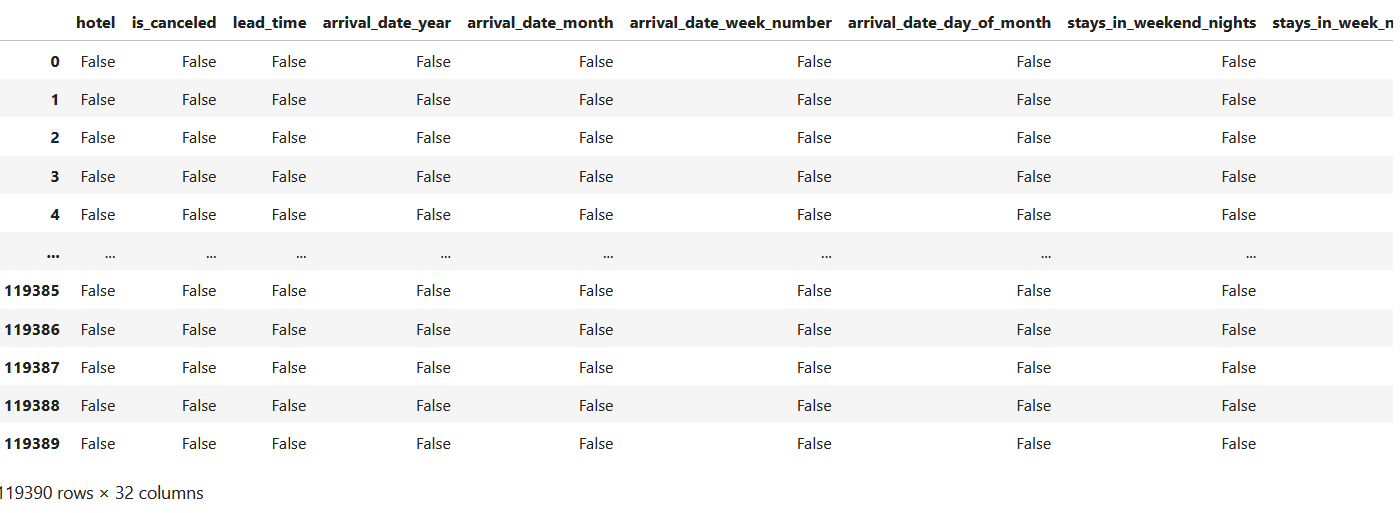
30 reservation\_status 119390 non-null object

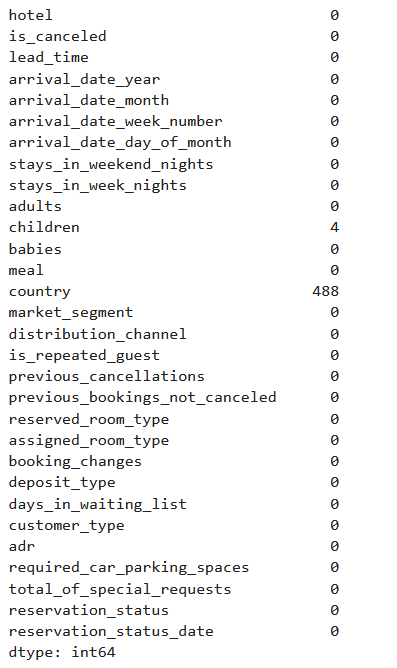
31 reservation\_status\_date 119390 non-null object

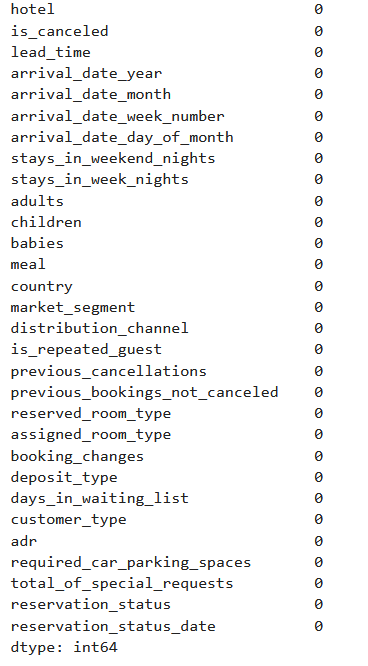
dtypes: float64(4), int64(16), object(12)

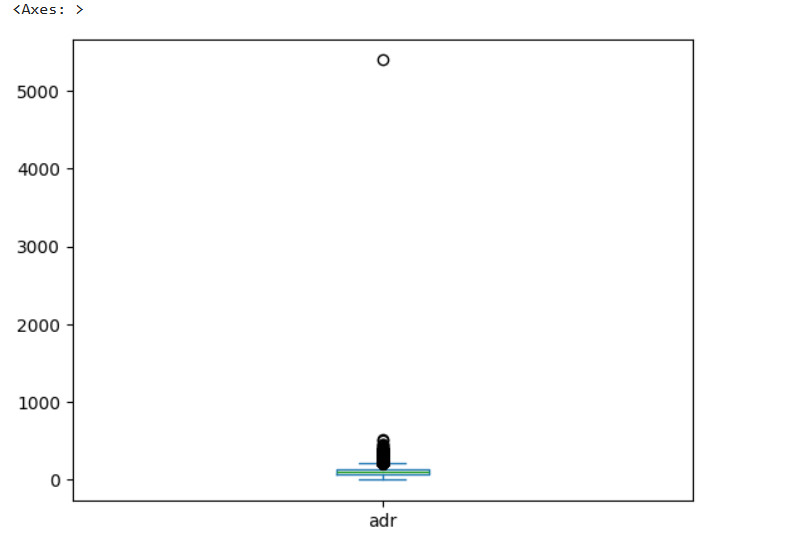
memory usage: 29.1+ MB

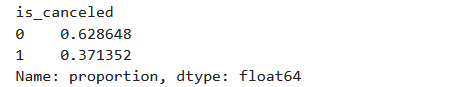


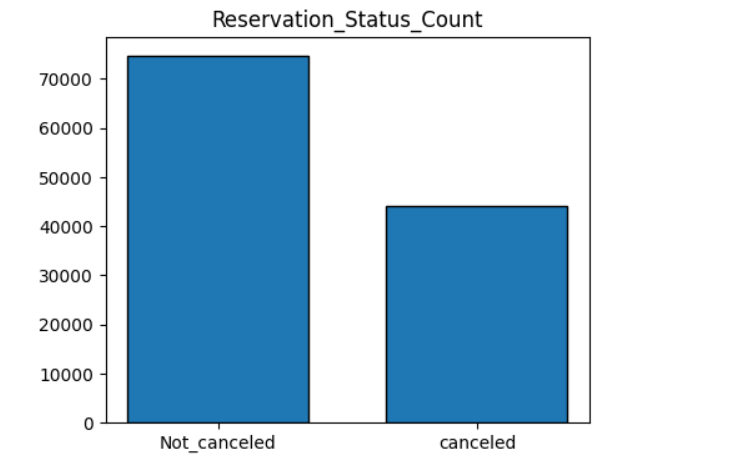


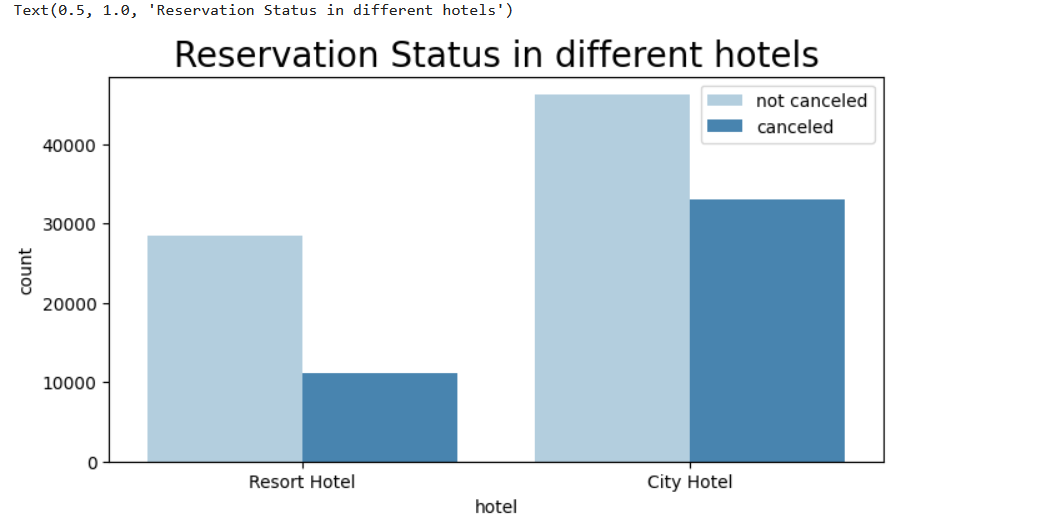


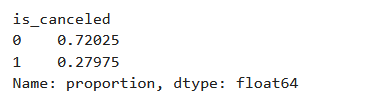


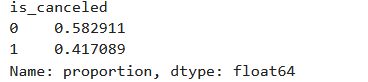


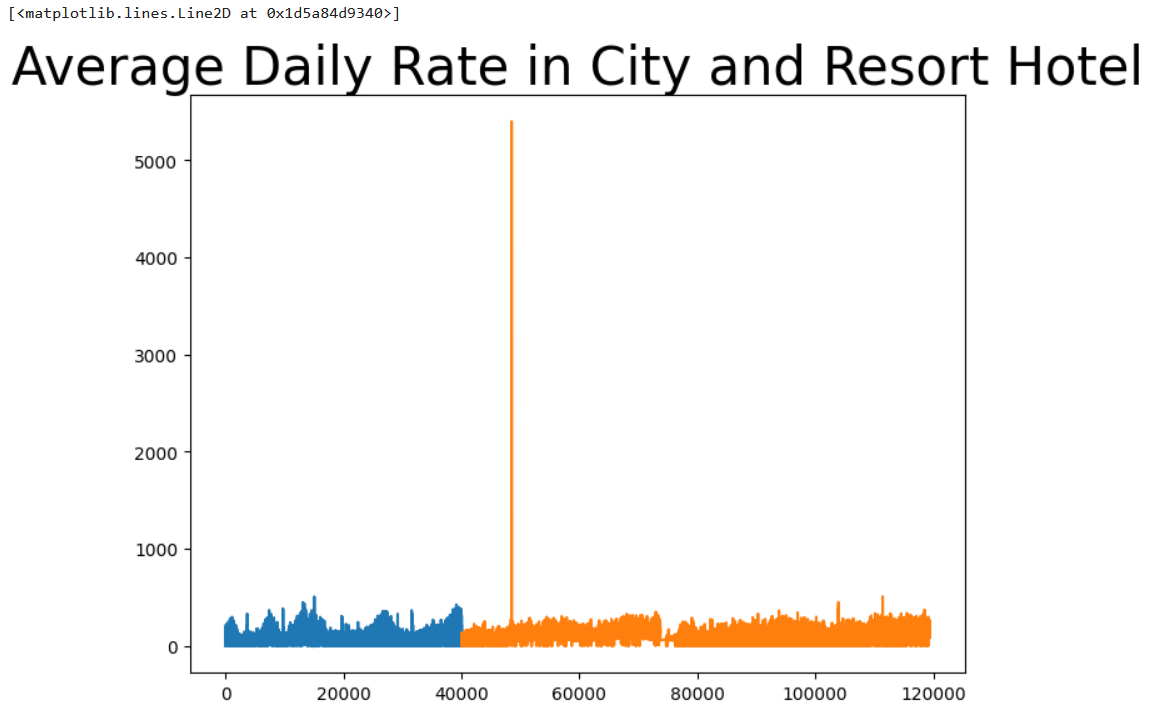


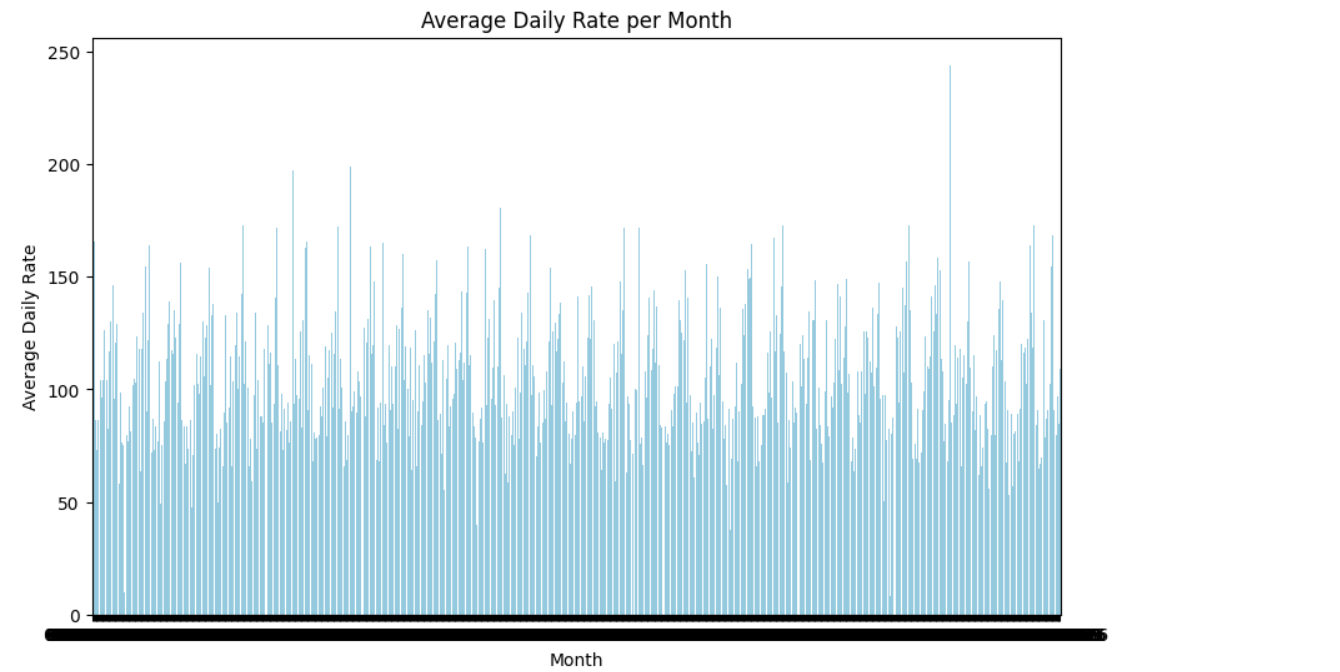




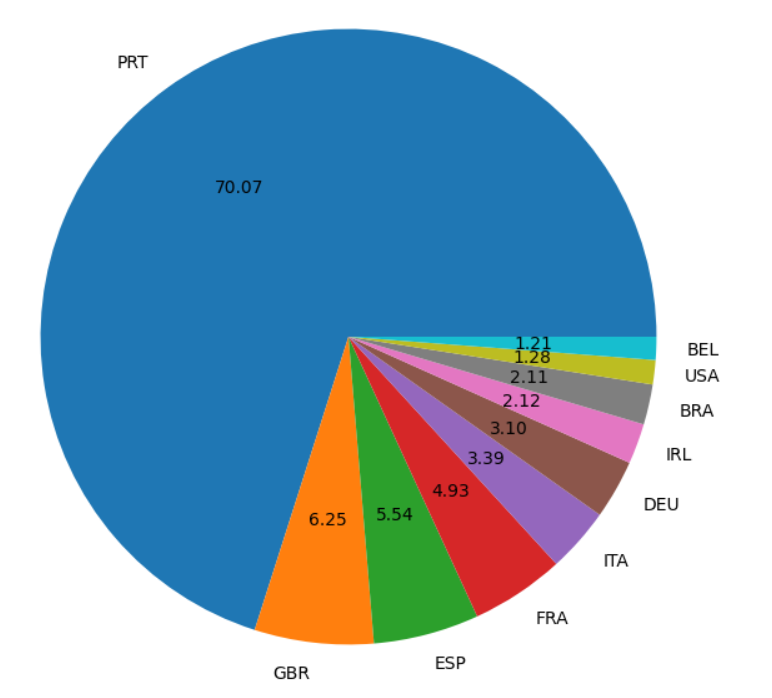




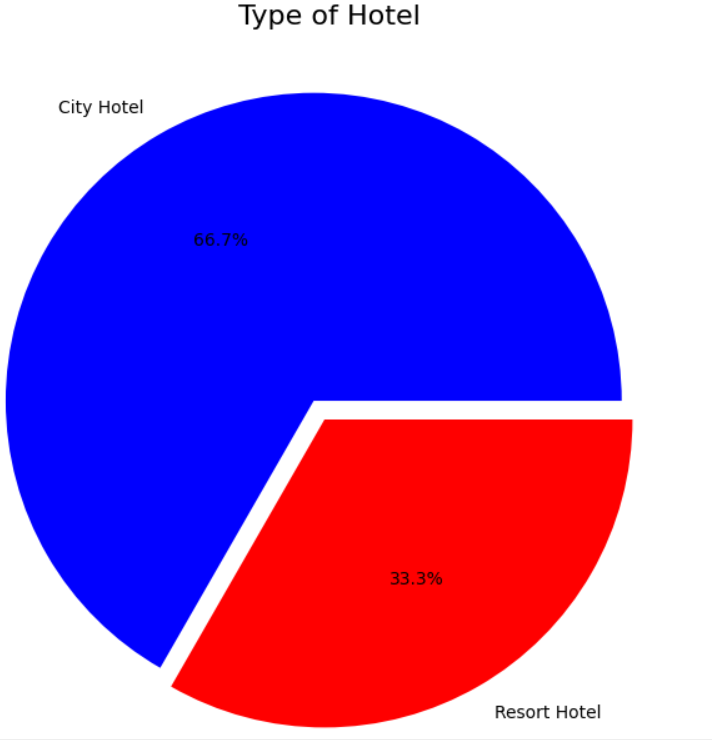


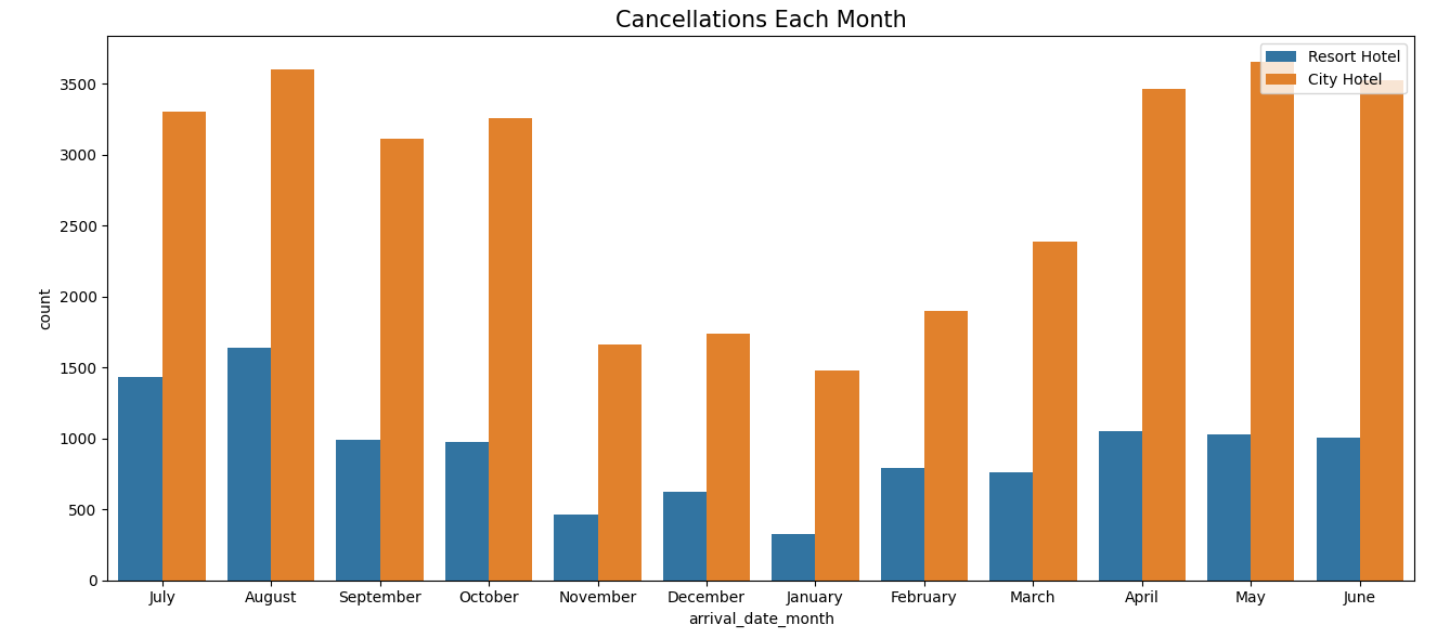


**Top 10 Countries with reservation cancelled**



Text(0.5, 1.0, ' Type of Hotel')





**CONCLUSION**

In data science, a hotel booking system provides a rich dataset that offers insights into various aspects of hotel operations and customer behavior. Analyzing such data can reveal booking patterns, financial metrics, and customer preferences, which are crucial for optimizing pricing strategies, improving customer experience, and making data-driven business decisions.

The hotel booking analysis project provides valuable insights into various patterns and trends within hotel reservation data. Through careful data preprocessing, exploration, and visualization, several key findings were uncovered. Firstly, a significant proportion of reservations were found to be canceled highlighting the importance of understanding customer behavior and potential issues leading to cancellations. Analyzing the data based on hotel types—Resort Hotel and City Hotel—revealed distinct booking and cancellation patterns, as well as differences in average daily rates (ADR). The ADR distribution and trends over time offered insights into pricing strategies and revenue management, while the visualization of seasonal trends, such as average daily rates per month and cancellations by month, underscored the impact of seasonality on bookings. Additionally, identifying the top 10 countries with the highest number of cancellations provided geographical insights that could help target specific markets to reduce cancellation rates. Finally, understanding the proportion of each hotel type offered an overview of the dataset composition, aiding in the interpretation of overall trends. These insights are crucial for hotel managers to make informed decisions, optimize operations, enhance revenue, and tailor marketing efforts to meet customer needs more effectively.

**REFERENCES**

* https://chatgpt.com/c/28a969c4-3422-424f-8314-9c29df4bc643
* blob:https://github.com/8d019540-b51d-4aa7-9a01-f615eae7bb81
* https://github.com/Kajal3322/Hotel-Booking-Analysis-Using-Python/blob/main/Hotel%20Booking%20Analysis%20Final\_version\_Group\_3\_MIS\_776\_KC.ipynb