Project Report
On

"HOTEL BOOKING CANCELLATION ANALYSIS USING PYTHON"

Submitted to the **Uttaranchal University** in partial fulfilment of the requirements for the award of the Degree of

BACHELOR OF COMPUTER APPLIC ATIONS

Submitted by

Tamash Jyoti Neog

(Learner ID: 2223020165)

Under the Guidance of

Ms. Apoorva S Shekhawat, AI Scientist – II (Artificial Intelligence), Infoedge India Ltd.

UTTARANCHAL UNIVERSITY, DEHRADUN

Acknowledgement

I am deeply grateful to all those who have contributed to the successful completion of this project.

First and foremost, I would like to express my sincere gratitude to my guide, Ms. Apoorva S Shekhawat, for their invaluable guidance, constant support, and encouragement throughout the duration of this project. Their expertise and constructive feedback have significantly contributed to shaping the direction of this research, and I am truly thankful for their patience and insights.

I would also like to extend my heartfelt thanks to **Centre for Distance and Online Education (CDOE), Uttaranchal University** for providing a conducive environment for learning and research. The resources and knowledge imparted have been instrumental in the successful completion of this project.

Declaration

I, Tamash Jyoti Neog, declare that the project titled 'Hotel Booking Cancellation Analysis Using Python' is an original work carried out by me under the guidance of Ms. Apoorva S Shekhawat. The work is not copied from any source, and no part of the project has been submitted elsewhere for any other degree.

Signature of Learner: Tamash Jyoti Neog

Name of Learner: Tamash Jyoti Neog

Learner -Id: **2223020165**

Certificate of Originality

This is to certify that the project titled 'Hotel Booking Cancellation Analysis Using Python' submitted by Tamash Jyoti Neog, 2223020165, in partial fulfillment of the requirements for the degree of Bachelor of Computer Applications, is an original work carried out under my supervision.

Apoorva S Shekhawat
AI Scientist – 2
Artificial Intelligence
Infoedge India Ltd.
12/06/2025

Table of Content

1. INTRODUCTION

The hotel and hospitality industry is a cornerstone of the global service economy, offering millions of bookings each year across city hotels, resorts, and vacation rentals. While the rise of online booking platforms has made hotel reservations more convenient for customers, it has also created operational challenges for hoteliers—most notably, the issue of **booking cancellations**.

Hotel booking cancellations result in **lost revenue, inaccurate forecasting, suboptimal resource allocation, and decreased operational efficiency**. In high-demand seasons, a cancellation may mean lost opportunities that cannot be recovered. If hotels fail to identify why and when customers cancel, it becomes difficult to adopt effective mitigation strategies. The growing trend of flexible refund policies and last-minute cancellations further complicates the issue, making it critical to explore cancellation behavior using data.

This project, titled **"Hotel Booking Cancellation Analysis Using Python"**, is centered around understanding and analyzing cancellation patterns using a real-world dataset of hotel bookings. The aim is to generate insights through data exploration, visualization, and pattern recognition to help hotels better understand customer behavior and minimize cancellation-related losses.

Purpose of the Project

This project investigates the **key variables** that contribute to hotel reservation cancellations using descriptive data analysis methods. The dataset used was obtained from **Kaggle** and contains **over 119,000 booking records** for both **resort hotels** and **city hotels**. Each record includes detailed information about the booking, such as:

- Booking status (canceled or not)
- Type of hotel
- Duration of stay
- Date of booking
- Booking channel (e.g., online travel agency, direct)

- Customer nationality
- Price per night (ADR – Average Daily Rate)
- Deposit type and special requests

The project uses **Python programming** to analyze these variables and discover relationships between them and cancellation behavior. All development work is performed in **Jupyter Notebook**, ensuring transparency and easy interpretation of the results through embedded visualizations.

Problem Significance

Cancellations are more than just an operational nuisance—they impact a hotel's ability to forecast demand, manage inventory, and plan staffing. Especially during high-demand seasons, a canceled booking often means lost revenue that cannot be reclaimed at short notice.

Unpredictable cancellations lead to:

- Empty rooms despite apparent full booking
- Wasted promotional efforts or discounts
- Overstaffing or understaffing
- Misaligned inventory, utilities, and services
- Reduced trust in booking platforms and customer segments

Many cancellations are influenced by:

- Customer behavior and preferences
- Booking source (Online Travel Agency vs. direct booking)
- Length of lead time between reservation and check-in
- Room pricing fluctuations and high ADR (Average Daily Rate)
- Presence or absence of special requests
- Seasonality and peak travel months

To illustrate the real-world context of the problem, the following image represents a typical hotel front desk scenario where guests check in or cancel their reservations in person.

Figure 1.1 – Hotel Front Desk at Check-in Time

Understanding how these factors interact allows hotels to fine-tune their cancellation policies, marketing strategies, and pricing models. This project provides a data-driven foundation for such decision-making and supports a shift from reactive to proactive hotel operations.

Tools and Technologies Used

The following open-source tools and technologies were used in this project:

- **Python** – For all scripting and data analysis logic
- **Pandas** – For data cleaning, transformation, and aggregation
- **NumPy** – For performing numerical computations
- **Matplotlib** – For generating basic visualizations and plots
- **Seaborn** – For styled statistical visualizations
- **Jupyter Notebook** – For code execution, output, and documentation in a unified environment

These tools provide a powerful ecosystem for data analysis and visualization, enabling reproducible research workflows and in-depth exploratory analysis.

Project Scope

The scope of this project is limited to **exploratory data analysis (EDA)** using a structured Python-based pipeline. The focus is on identifying observable trends in the data rather than building predictive models.

The following areas are covered:

- Comparative analysis of **cancellations by hotel type**
- Trends in **cancellations across months** (seasonality)
- Effect of **ADR (price)** on cancellation rates
- **Country-wise distribution** of cancellations
- Role of **lead time and deposit type**
- Influence of **market segment** (e.g., OTA, direct bookings)
- Relationship between **special requests** and cancellations

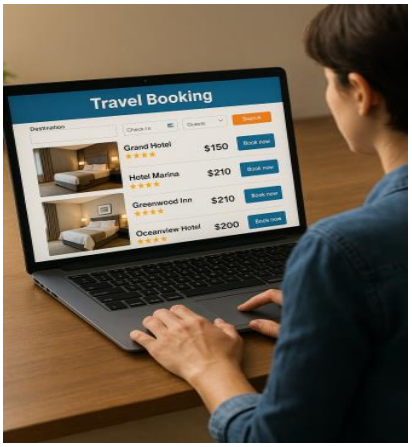

Figure 1.2 – Online Hotel Booking Interface Used by Customers

This image highlights the ease and flexibility of online hotel bookings, which also increases the risk of cancellations. Understanding this behavior is essential to the project's goal of analyzing trends and helping hotels adapt their strategies accordingly.

Academic Value

From an academic standpoint, this project serves to:

- Strengthen Python programming and data handling skills
- Apply data cleaning, filtering, and aggregation methods
- Gain experience in visual storytelling using charts and plots
- Interpret real-world datasets to derive business insights
- Document a full project using structured formatting, markdown, and visual outputs

The project aligns with the academic goals of the BCA curriculum by demonstrating the integration of programming, statistics, and real-world problem-solving.

Business Relevance

On the business front, the insights generated by this analysis can help hotel managers:

- Implement better **overbooking strategies**
- Identify **high-risk booking sources**
- Restructure **cancellation policies** during specific seasons
- Focus on **reliable market segments**
- Plan for **seasonal staffing and inventory management**

By grounding decision-making in data, hotels can increase profitability while offering better guest experiences.

Summary

This chapter introduced the problem of hotel booking cancellations, the tools and technologies used to explore it, and the structure of the project. By using Python-based data analysis techniques in Jupyter Notebook, the project aims to uncover patterns and behaviors behind cancellations and present actionable findings. The subsequent chapters will cover the project objectives, system analysis, software and hardware specifications, system design, testing, and reporting.

2. OBJECTIVE

The goal of this project, titled **"Hotel Booking Cancellation Analysis Using Python"**, is to analyze a real-world hotel booking dataset in order to understand the patterns and behaviors associated with reservation cancellations. The project seeks to identify critical factors that influence customer cancellation decisions, such as seasonality, pricing, booking source, and customer characteristics. These insights are intended to help hotel businesses reduce cancellation rates, improve occupancy forecasting, and design data-driven business strategies.

The analysis is conducted using Python within the Jupyter Notebook environment. The dataset, sourced from Kaggle, contains over 119,000 booking records from both resort and city hotels. Through structured data cleaning, filtering, and visual analysis, this project transforms raw booking records into meaningful insights that can directly support operational and financial decisions.

This analysis aims not only to interpret the cancellation behavior statistically but also to present it in a way that is easy to understand for decision-makers in the hospitality industry. The insights gained through this project can be translated into strategies that reduce avoidable cancellations, optimize hotel occupancy, and improve overall guest satisfaction. With an increasing reliance on data across industries, the hospitality sector too must evolve by adopting analytics to solve its operational challenges.

Purpose of the Project

The primary purpose of this project is to:

- Understand which variables are associated with higher booking cancellation rates.
- Compare cancellation trends across different types of hotels.
- Explore the impact of lead time, ADR (Average Daily Rate), deposit type, and booking channels on cancellations.
- Investigate seasonality and identify months with peak cancellation activity.

- Discover behavioral differences between customer groups based on market segment and nationality.

These objectives form the foundation of the project and guide every phase of the analysis. By understanding what drives cancellations, the project helps turn raw data into insights that hotels can act upon to improve operations and customer satisfaction.

By meeting these goals, the project not only helps quantify cancellation behavior but also assists in identifying actionable trends. It encourages the use of real historical data as a guide for future planning, enhancing both the predictive and preventive capabilities of hotel management teams.

Key Objectives

- **To identify patterns in booking cancellations**
  Examine which types of bookings are more likely to be canceled based on available attributes like hotel type, ADR, lead time, etc.
- **To clean and preprocess the dataset for reliable analysis**
  Handle missing values, remove outliers, and convert data types to ensure high data quality and consistency.
- **To perform detailed exploratory data analysis (EDA)**
  Use statistical summaries and visualizations to explore relationships between variables and detect trends.
- **To visualize booking behavior and cancellation trends**
  Create clear and informative plots that explain the impact of various factors on cancellation probability.
- **To help hotel businesses make data-driven decisions**
  Convert analytical insights into practical recommendations for operations, policy, and pricing.
- **To ensure the analysis is replicable and well-documented**
  Maintain all steps within Jupyter Notebook to allow easy understanding, traceability, and reusability of the project.

Each objective listed above contributes to the overall structure of the Jupyter Notebook, with separate code cells and sections designed to fulfill one or more goals. The process of meeting these objectives also served as a roadmap for implementing clean, readable, and modular code that supports scalability and ease of understanding.

Broader Impact

While this project is developed as part of an academic exercise, its findings have real-world implications. Hoteliers, revenue managers, and business analysts can use similar analyses to:

- Improve guest retention strategies.
- Develop more restrictive or flexible refund policies based on data trends.
- Better manage overbooking policies and dynamic pricing.
- Predict and mitigate high-cancellation risk periods.

Additionally, this kind of data analysis can support broader organizational changes. It encourages hotels to maintain cleaner datasets, adopt more integrated reservation systems, and explore the use of analytics dashboards in their daily operations. The techniques used in this project can also be extended to other areas such as guest satisfaction analysis, pricing optimization, and seasonal demand forecasting.

Educational Objectives

This project also serves as a valuable academic learning experience by enabling:

- Real-world application of Python programming and libraries
- Mastery of Jupyter Notebook for technical reporting
- Exposure to real data analysis workflows
- Development of structured, professional-quality documentation
- Enhancement of problem-solving and business interpretation skills

By working on this project, I was able to apply academic knowledge in a practical setting. It helped me strengthen both my technical programming capabilities and my ability to interpret real-world data. I learned how to present findings clearly through visualization and documentation, making the results accessible even to non-technical stakeholders. Overall, this experience has significantly improved my confidence in conducting data-driven investigations.

Summary

The primary and secondary objectives outlined above guide the entire structure and flow of this project. From data collection and preprocessing to visual analysis and insights, each step is taken to fulfill the overarching aim: to understand and minimize hotel booking cancellations through analytical exploration.

This project not only fulfills an academic requirement but also reflects real-world applications of data science in hospitality. By focusing on practical implementation and insight generation, it bridges the gap between classroom learning and business problem-solving. The clearly defined objectives ensure that the analysis remains goal-oriented and relevant throughout the project lifecycle.

3. SYSTEM ANALYSIS

System analysis is the process of examining a proposed project or problem domain in-depth, identifying the core issues, understanding stakeholder expectations, and defining the technical framework for designing a viable system. For this project — **"Hotel Booking Cancellation Analysis Using Python"** — system analysis plays a foundational role. It transforms the real-world business problem of excessive booking cancellations into a structured data analysis model that helps uncover hidden patterns, explain cancellation behavior, and offer practical suggestions based on data.

Unlike traditional software systems that offer a user interface or transactional capabilities, the system in this project is an **analytical engine**, built entirely using **Python in Jupyter Notebook**, and powered by structured hotel reservation data. The "system" refers to the complete pipeline — from data loading and cleaning to exploratory data analysis (EDA), visualization, and final business interpretation.

In this chapter, we define the need for such a system in today's hotel business environment. We also explore the steps taken in preliminary investigations, feasibility evaluation, and planning that shaped the development of this solution.

3.1 IDENTIFICATION OF NEED

The core motivation behind this project stems from a real and pressing issue in the hotel industry — the growing rate of **booking cancellations**. While digital transformation has made it easier for customers to book hotels from anywhere in the world, it has also introduced operational complications for hotels, especially in terms of cancellations that occur too frequently or too close to the date of arrival.

Most hotels today rely heavily on **online travel agencies (OTAs)** like Booking.com, Expedia, and Agoda to receive bookings. These platforms allow customers to cancel with little or no penalty. While this policy increases customer flexibility, it poses a major challenge to hotels trying to maintain optimal occupancy and resource allocation.

Let us break down **why there is a strong need** for this analytical system.

High Cancellation Rates Hurt Revenue

Each cancellation represents a potential revenue loss. Even though another booking may replace it later, hotels often lose pricing leverage when rooms are rebooked at the last minute or remain vacant. When cancellations are frequent, they lead to:

- **Revenue instability**
- **Poor forecasting accuracy**
- **Wasted promotional or discount efforts**
- **Over- or under-staffing**

Manual Forecasting Is No Longer Sufficient

Traditionally, hotels have forecasted occupancy using:

- Historical seasonal data
- Local event calendars
- Basic trend observation

However, this method is **too generic**. It does not account for changing customer behavior or external variables like:

- Lead time (time between booking and check-in)
- Special requests
- Booking source (OTA, group, corporate)
- Customer nationality
- Deposit type

Without data analytics, such variables remain invisible to hotel managers — resulting in **missed opportunities to predict and prevent cancellations**.

Data Is Available — But Underutilized

Hotels now have access to **rich booking data** through property management systems (PMS), OTAs, and direct booking platforms. This data includes:

- Guest details
- Booking timestamps
- Price and rate plans
- Length of stay
- Cancellations and modifications

However, **many hotel operators do not analyze this data effectively**, either due to lack of expertise, time, or tools. This creates a significant gap between data collection and insight generation — a gap this project aims to fill.

Python-Based EDA Can Deliver Valuable Insights

The analysis system in this project is built with:

- **Python** for scripting logic
- **Pandas** for data manipulation
- **Matplotlib & Seaborn** for visualizations
- **Jupyter Notebook** for integrating code, charts, and documentation

This technology stack allows for a **modular, reusable, and highly visual analytical workflow** that produces actionable insights such as:

- Which types of hotels (city vs. resort) face more cancellations
- Which months or seasons are risk-prone
- How lead time and price influence cancellations
- Which customer profiles are more likely to cancel

All this can be derived **without writing advanced machine learning models**, making it cost-effective, scalable, and accessible.

Current Industry Gap Justifies This Project

Most hotels, especially in budget and mid-range segments, do not have in-house data science teams. They rely on limited reports generated by booking engines or property management systems, which:

- Do not provide visual analytics
- Do not explain behavior — only trends
- Do not show comparisons across hotels or time periods

By building this project using freely available datasets (from Kaggle) and open-source tools, the student demonstrates how a **BCA-level technical skill set** can address a real business problem effectively.

Business Goals That Define the Need

This project system is needed not just for technical exploration, but to serve the following **practical business goals**:

Business Goal	How This Project Helps
Reduce cancellation-related revenue loss	By analyzing past patterns and risky booking profiles
Improve occupancy forecasting	By identifying seasonal and behavioral trends
Target more reliable customer segments	Through nationality and market segment-based analysis
Design better pricing & deposit policies	By understanding the effect of ADR and deposit types
Enhance operational planning	By correlating lead time, stay duration, and special requests

Table 3.1 – Business Objectives and How the Project Addresses Them

Conclusion of Need Analysis

The **need for this system is clear**: the hotel industry is experiencing rising cancellation rates due to the shift toward digital booking and customer-friendly policies. To remain competitive, hotels must transition from **instinct-based decisions to insight-driven strategies**. This project is a step in that direction.

By offering a self-contained, offline, and easy-to-run analysis system based on open data and Python, the project not only addresses a real business issue but also demonstrates how **data science can serve practical goals** without high costs or complexity.

3.2 PRELIMINARY INVESTIGATION

Before developing any analytical solution, it is essential to perform a **preliminary investigation** to understand the problem domain, validate the relevance of the project, identify suitable datasets, and select the correct tools and technologies. This phase ensures that the project is **technically aligned, practically feasible, and strategically valuable**.

In the case of this project — **"Hotel Booking Cancellation Analysis Using Python"** — the preliminary investigation was focused on understanding the growing challenge of hotel booking cancellations, the factors contributing to this problem, and how a data analysis approach could be used to extract useful patterns from historical booking records.

Understanding the Business Domain

The hospitality industry has become increasingly reliant on **online booking systems**. Platforms like Booking.com, Agoda, and Expedia allow users to:

- Reserve rooms instantly
- Cancel bookings without penalties
- Compare prices across dozens of hotels

- Book multiple options and cancel later based on convenience

From a customer standpoint, this is efficient and flexible. But for hotel operators, this shift has **amplified uncertainty** in occupancy management. Especially in high-traffic tourist areas, where hotels may expect high occupancy, **last-minute cancellations lead to lost revenue and wasted resources**.

Several questions arise:

- Why do guests cancel?
- Are some months or customer groups more prone to cancel?
- Do pricing, deposits, or booking channels influence cancellation rates?

These questions helped define the need to investigate the issue further using **real data** rather than relying on assumptions.

Dataset Search and Selection

Once the problem was defined, the next step was to search for a relevant dataset. After exploring various sources, the dataset titled **"Hotel Booking Demand"** available on **Kaggle** was selected for the following reasons:

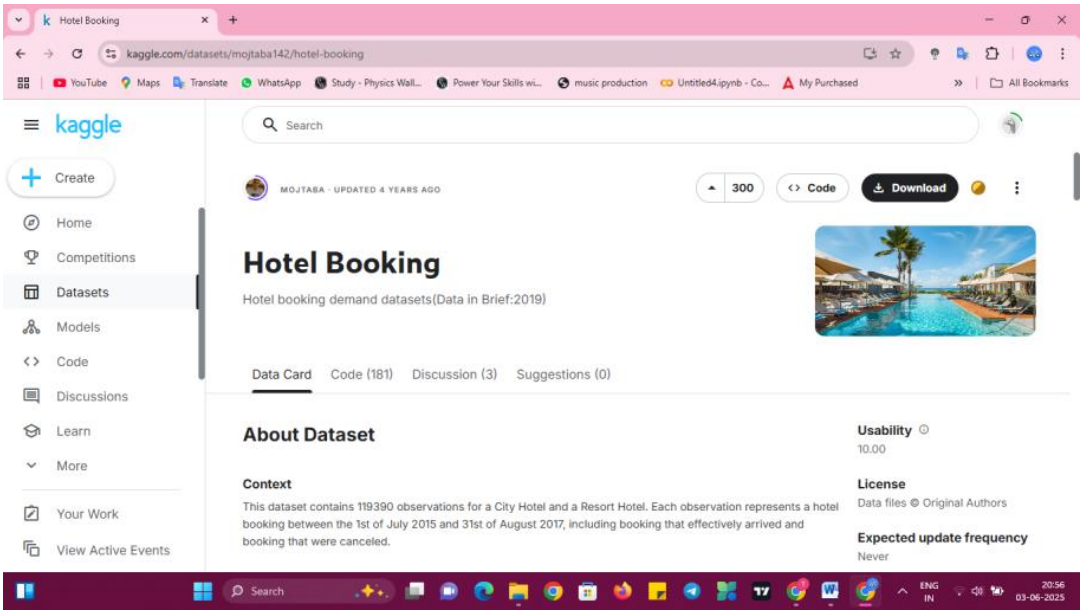

Figure 3.1 – Source: Hotel Booking Demand Dataset on Kaggle

- Contains over **119,390 records** — offering large enough volume for meaningful analysis.
- Covers **two hotel types**: **City Hotel** and **Resort Hotel** — allowing comparative study.
- Includes detailed attributes such as:
    - Whether the booking was **canceled** (is_canceled)
    - **Lead time** (days between booking and arrival)
    - **Market segment** (OTA, Direct, Groups)
    - **ADR** (Average Daily Rate per night)
    - **Deposit type**
    - **Special requests**, number of guests, stay duration
    - **Country** of origin and customer demographics
- Dataset is **anonymized and publicly available**, ensuring it can be used for academic analysis without any ethical issues.

Exploring the Dataset Features

The following important columns in table 3.2 were identified during initial dataset review:

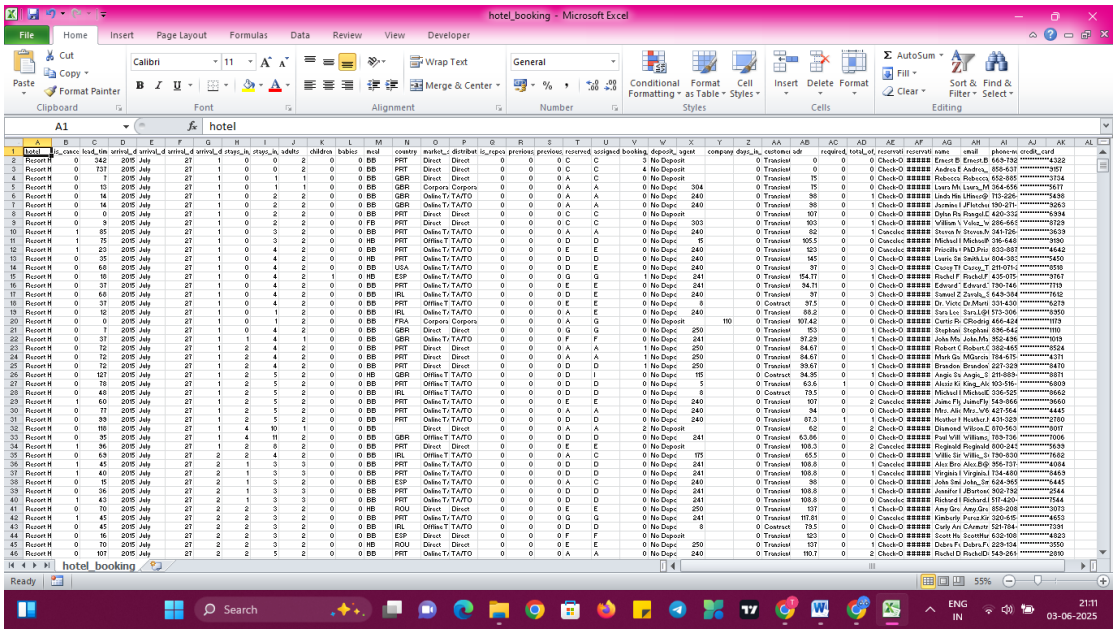

Figure 3.2 – Sample Excel View of the Dataset Before Analysis

Field Name	Description
is_canceled	Target variable (1 = canceled, 0 = not canceled)
hotel	Hotel type (City or Resort)
lead_time	Number of days before arrival the booking was made
adr	Average Daily Rate (price per night)
deposit_type	Type of deposit (No Deposit, Non Refund, Refundable)
market_segment	Source of booking (Online TA, Direct, Corporate, Groups)
reservation_status	Final status (Check-Out, No-Show, Canceled)
arrival_date_month	Month of arrival — useful for seasonality analysis
customer_type	Type of customer (e.g., Transient, Group, Contract)
country	Guest's country of origin

Table 3.2 – Key Attributes in the Hotel Booking Demand Dataset

These features made it possible to explore **multiple dimensions of customer behavior** and cancellation trends.

Technology Selection and Justification

Based on the dataset format (.csv) and the nature of the task (exploratory data analysis and visualization), **Python** was selected as the development language. Python provides a wide array of libraries tailored for data science:

Tool/Library	Purpose in the Project
Python 3.8+	Base language for scripting and project logic
Jupyter Notebook	Environment for code, output, and explanation in one interface
Pandas	Reading, cleaning, manipulating structured data (DataFrames)
NumPy	Supporting numerical operations and array processing
Matplotlib	Plotting line graphs, bar charts, and visual summaries
Seaborn	Creating attractive, informative statistical visualizations

Table 3.3 – Technology Stack Used in the Project

These tools are:

- **Free and open-source**
- **Platform-independent**
- Easy to install using **Anaconda Navigator**, which bundles all dependencies

This stack was chosen after comparing alternatives like R (good but steeper learning curve) and Excel (user-friendly but limited for large datasets and statistical analysis).

Investigating Project Direction

In this phase, exploratory activities were conducted to validate:

- Whether the dataset is clean and suitable for analysis
- Whether cancellation behavior varies across hotel types
- If seasonal trends are visible in the cancellation pattern
- Whether ADR, deposit type, or lead time correlates with cancellations

Initial exploratory queries confirmed:

- City hotels have higher cancellation rates than resort hotels.
- Bookings with longer lead times and “no deposit” policies are more likely to be canceled.
- Cancellations peak in summer months (August, July) due to uncertain travel plans.

These early insights confirmed that the dataset could deliver **rich, business-relevant patterns** if analyzed systematically. This justified continuing to the system design and implementation phase.

Academic and Practical Alignment

This investigation phase also confirmed that the project aligns with:

- The academic goals of the BCA curriculum
- Real-world problems faced by hospitality businesses
- Practical use of Python programming and visual analytics
- Ethical data usage (no personal or sensitive data involved)

The result is a project that is both educational and impactful.

Conclusion of Preliminary Investigation

The preliminary investigation laid the foundation for this project by confirming that:

- The **problem** is relevant, realistic, and solvable
- A **suitable dataset** is available for open academic use
- The **tools and technologies** are freely accessible, powerful, and platform-independent
- The **direction of analysis** supports both learning outcomes and real-world application

This phase ensured that the system design would proceed confidently, knowing that all core components — data, tools, goals, and scope — are validated.

3.3 FEASIBILITY STUDY

Before any system is developed or implemented, it is essential to evaluate whether the proposed solution is feasible in real-world scenarios. A **feasibility study** helps to assess whether a system is **technically sound**, **operationally viable**, and **economically justified**. It forms the foundation for decision-making, especially in academic and industry-oriented projects where resources, time, and expected outcomes must align.

In the case of this project, titled **"Hotel Booking Cancellation Analysis Using Python,"** the feasibility study was conducted with the objective of ensuring that the system would perform effectively using available tools, function efficiently in its

intended environment, and deliver value without incurring significant cost or complexity.

Unlike traditional software systems which involve full-stack development, front-end design, database integration, and deployment, this project focuses on **exploratory data analysis (EDA)** using structured datasets. Hence, the nature of feasibility assessment is different — it places greater emphasis on **analytical performance**, **computational reliability**, and **interpretability of outputs** rather than user interface functionality or deployment logistics.

The feasibility of the project is examined across the following three dimensions:

- **Technical Feasibility:**
  To evaluate whether the available tools, platforms, and hardware are sufficient to implement the solution efficiently.
- **Operational Feasibility:**
  To assess how well the system performs its intended function, and whether the final output meets academic and analytical expectations.
- **Economic Feasibility:**
  To ensure that the system can be developed and used with minimal financial expenditure, relying only on freely available resources.

The subsections below explore each of these aspects in detail, beginning with technical feasibility — the most fundamental criterion for launching the project.

3.3.1 TECHNICAL FEASIBILITY

Technical feasibility examines whether the technical resources and environment available are capable of supporting the project's development, execution, and maintenance. It looks at the **tools used**, the **capability of the hardware**, the **volume and structure of data**, and the **ease of working with the selected technologies**.

Since this project was conducted in a **local academic setup** using **Python and Jupyter Notebook**, the feasibility was focused on data handling performance, memory efficiency, and visual rendering of insights.

Local System and Environment

The entire analysis was performed on a standard personal laptop with no specialized hardware or enterprise-level infrastructure. The specifications of the system are as follows:

Component	Specification
Processor	Intel Core i3, 2.3 GHz
RAM	4 GB DDR4
Storage	2 GB free space used for project files
Operating System	Windows 10 (64-bit)
Python Version	Python 3.8
Environment Used	Anaconda Navigator with Jupyter Notebook

Table 3.1 – Minimum Hardware and Software Environment Used in the Project

This configuration was sufficient to load, clean, process, and visualize the entire dataset of over **119,000 records**, demonstrating the technical compatibility and efficiency of the system setup.

Tools and Technology Stack

The system relies entirely on **open-source, platform-independent software**. The key components of the stack include:

Tool/Library	Function in the Project
Python	Primary programming and scripting language
Jupyter Notebook	Code execution and inline visualization, markdown-based documentation
Pandas	Data wrangling, filtering, transformation, and grouping
NumPy	Supporting matrix operations and numerical calculations
Matplotlib	Creating basic visual plots (bar, line, pie charts)
Seaborn	Generating high-level, statistical and comparative visualizations

Table 3.2 – Python Libraries and Their Purpose in the Project

All these libraries were installed once through **Anaconda**, making the setup process easy and manageable even for users with limited technical experience.

Dataset Processing Capability

The dataset — titled **"Hotel Booking Demand"** — was downloaded from Kaggle and imported into the project using Pandas. It contains **119,390 rows and 32 columns**, including:

- Booking status (canceled / not canceled)
- Hotel type (City / Resort)
- Average Daily Rate (ADR)
- Lead time
- Deposit type
- Market segment
- Nationality
- Arrival date (year, month)

Despite its volume and variety, the dataset was processed effectively using filtering, grouping, and plotting operations without any performance lag or memory issues.

Each part of the dataset was explored through summary statistics, correlation heatmaps, grouped aggregation, and filtered plotting. The Python libraries handled all transformations within a few seconds on the given hardware, making the system technically efficient.

Code Execution and Structure

The entire project was written in a single **Jupyter Notebook (.ipynb)**, structured into distinct functional sections:

- **Dataset Import and Preview**
- **Missing Value Detection and Handling**
- **Data Type Conversion**
- **Exploratory Data Analysis (EDA)**
- **Visualizations and Plots**
- **Insights and Observations**

Each section was preceded by markdown headers, and outputs were shown inline. This modular structure helped:

- Avoid confusion during code execution
- Debug and test smaller parts individually
- Ensure transparency and traceability for academic evaluators

Additionally, each graph or analysis result was accompanied by text interpretation to make the outcome readable and business-friendly.

Platform Independence

All components of this project are **OS-independent**, meaning the same code and data can be executed:

- On Windows, macOS, or Linux
- With any Python version >= 3.8

- Using any modern browser to access Jupyter Notebook

This enhances both **portability** and **reproducibility**, which are key technical strengths for academic projects.

Challenges Encountered and Solutions Applied

Challenge Faced	How It Was Resolved
Large CSV file slowed initial load	Used low_memory=False and limited columns on first load
Missing values in key columns	Replaced or dropped intelligently based on context
Date/time fields not recognized	Converted using pd.to_datetime() and extracted month/year manually
Long runtime for grouped plots	Pre-grouped data and cached common groupings before visualization

Table 3.3 – Technical Issues Faced and Their Solutions

These small issues were resolved with basic coding strategies, without needing external help or high-end computing, reaffirming the technical ease of project execution.

Scalability and Extension Potential

Although designed as an academic project, the technical model allows for future upgrades such as:

- Integration with new datasets from hotel chains
- Adding predictive models (e.g., logistic regression for cancellation prediction)
- Exporting visuals into dashboards or reports
- Connecting the analysis with a basic frontend using Streamlit (if needed)

This makes the system technically **scalable and extendable**, even if it was built for offline, one-time execution.

Conclusion

The project is **technically feasible**, confirmed by the following:

- All tools used are open-source, well-documented, and appropriate for the task
- The dataset was handled smoothly despite its size
- The code was executed entirely on a standard local machine
- Jupyter Notebook provided the ideal environment for development and documentation

The system required no network setup, no security configuration, and no paid software — all while producing reliable and insightful analytical results.

3.3.2 OPERATIONAL FEASIBILITY

Operational feasibility determines whether a system will operate effectively in its intended environment and whether its outcomes will be usable, accessible, and understandable by the target audience. In the case of this project — **"Hotel Booking Cancellation Analysis Using Python"** — the primary environment is an academic research and evaluation setting, with additional value for business analysts in the hospitality industry.

The project does not rely on complex infrastructure, external APIs, or web interfaces. Instead, it focuses on **data processing and insight generation** within a local notebook interface, allowing for streamlined, efficient operation without external dependencies.

Execution Environment

The project is developed entirely in **Jupyter Notebook**, hosted locally using **Anaconda Navigator**. All data loading, cleaning, processing, and visualization tasks are structured as modular code blocks. Each code cell is supported by markdown commentary, ensuring clarity and traceability of every step.

Features of the execution model:

- Runs offline after setup
- Displays charts inline with code
- Does not require network connectivity or database access
- Easily re-runnable on any system with Python and Jupyter installed

This makes the project highly feasible for day-to-day academic and practical use.

Usability and Accessibility

The system prioritizes:

- **Ease of understanding** through markdown annotations
- **Sequential execution** from top to bottom
- **Visual interpretation** through labeled and well-formatted charts
- **Low technical dependency**, requiring only basic Python knowledge

All data operations — such as filtering, grouping, plotting — are encapsulated in readable code cells, designed to be understood even by beginners in Python or analytics.

Additionally, visualizations such as bar charts, pie charts, and heatmaps allow for immediate understanding of cancellation behavior trends, making the system operationally beneficial even for hospitality professionals unfamiliar with coding.

Adaptability and Reusability

The Jupyter Notebook is developed in a **modular and flexible manner**:

- The dataset can be replaced with any similar CSV file
- Filters and groupings can be edited with minimal effort
- Graphs and analysis modules can be reused in future projects

The entire workflow is organized to support **easy adaptation**, making this system not just a one-time tool, but a **template for future hotel analytics projects**.

Academic and Business Impact

From an academic standpoint, the system helps the student demonstrate:

- Real-world application of Python programming
- Structured thinking and modular programming habits
- Interpretation of datasets through both numbers and visuals

From a business perspective, the output can help:

- Identify peak cancellation periods
- Analyze deposit-type effectiveness
- Plan seasonal promotions based on lead times and booking trends

Thus, the system is not only operable but also **highly functional and insightful**.

Conclusion

The project is **operationally feasible** because:

- It runs in a self-contained, low-resource environment
- It supports non-technical users through clear documentation
- It is reusable for other hotel-related datasets
- It serves academic, analytical, and business purposes effectively

The combination of ease, transparency, and interpretability makes this system well-suited for continued usage and academic evaluation.

3.3.3 ECONOMIC FEASIBILITY

Economic feasibility assesses the cost-effectiveness of implementing the system. For academic data analysis projects, this means evaluating the **financial cost**, **time investment**, and **value of learning**. This project was intentionally designed to be executed at **zero monetary cost**, using **freely available tools and data**, completed within **10 days**, as declared in the original synopsis.

Software and Tool Cost

All tools used in the project are open-source and freely available. No subscriptions, licenses, or premium versions were required.

Resource/Tool	Cost
Dataset from Kaggle	Free
Python & Libraries	Free (Open-source)
Anaconda & Jupyter Notebook	Free
Excel (for dataset preview)	Already available
Internet (for download)	Minimal (one-time only)

Table 3.4 – Software Tools and Resources Used with Cost Summary

---

Hardware Requirements and Cost

The project ran smoothly on a **standard personal laptop** with the following:

- Intel Core i3 processor
- 4 GB RAM
- Windows 10 OS

No additional hardware purchases or upgrades were required. The cost of electricity and internet usage was negligible and typical for academic use.

Time and Effort Investment (10-Day Plan)

Based on the planning and execution strategy shared in the synopsis, the full project was completed in **10 days**, distributed as follows:

Task	Duration
Problem Identification	Day 1
Data Collection & Cleaning	Day 2–3
Exploratory Data Analysis	Day 4–6
Visualization Development	Day 7–8
Insight Generation & Reporting	Day 9–10

Table 3.5 – Compressed Project Timeline Based on 10-Day Execution Plan

This focused timeline ensured that all essential phases — from dataset inspection to reporting — were completed efficiently without compromising quality.

Return on Investment

While no direct monetary investment was made, the returns include:

- Strengthening of real-world analytical skills
- Creation of a reusable analytics notebook
- Enhanced understanding of hospitality trends
- Academic credit toward the BCA degree
- Foundation for future advanced projects (e.g., prediction, dashboards)

The system also has **career value** as a demonstrable portfolio piece for internships or entry-level analyst roles.

Conclusion

The project is **economically feasible** due to:

- Zero financial expenditure on tools or data
- Usage of existing hardware and minimal utilities
- Completion within the 10-day timeframe
- Educational and professional value exceeding cost

It is a model example of a **cost-efficient, high-value academic project**, ideal for students and learners with limited resources.

3.4 PROJECT PLANNING

Effective project planning is essential to ensure timely and successful completion of any technical or analytical task. For the project titled *"Hotel Booking Cancellation Analysis Using Python,"* the planning phase played a vital role in organizing the workflow, optimizing time allocation, and maintaining a clear direction throughout the development lifecycle. Unlike traditional software applications, this project revolves around data analysis, requiring a well-structured approach to dataset handling, exploratory investigation, visual interpretation, and final documentation.

The planning phase began with identifying all major tasks and sequencing them logically over a period of 10 days, which was the timeframe proposed in the original project synopsis. Each stage of the analysis was broken down into individual objectives and linked to specific deliverables to ensure that no critical step was missed.

The key focus areas of this project include:

- Understanding the real-world business problem of hotel booking cancellations
- Exploring the dataset in detail to assess structure, size, and missing values
- Cleaning the dataset to improve quality and consistency for analysis
- Conducting a comprehensive exploratory data analysis (EDA)
- Visualizing cancellation trends using Python libraries such as Matplotlib and Seaborn
- Interpreting patterns and drawing conclusions for business relevance
- Preparing a well-structured and formatted academic report using MS Word

Each of these components was assigned specific time and attention within the plan to maintain a smooth and systematic workflow.

Project Workflow Approach:

The project followed a linear yet modular development style. Each phase depended on the successful completion of the previous one, while also allowing room for revisiting earlier steps if issues or insights emerged. For example, after generating visualizations, it was sometimes necessary to go back and reprocess the dataset or apply filters to enhance clarity. This flexible yet structured approach ensured both consistency and quality.

Planning and Execution Sequence:

A day-wise timeline was created to provide a focused direction for execution. The work distribution over 10 days is shown below:

Day	Planned Activity	Description
Day 1	Topic Finalization and Requirement Study	Identified the hotel industry problem, finalized the project title, reviewed cancellation trends and their impact on business operations.
Day 2	Dataset Collection and Initial Review	Downloaded the *Hotel Booking Demand* dataset from Kaggle. Conducted initial reviews using df.head(), df.info() and inspected structure and data quality.
Day 3	Data Cleaning and Preprocessing	Cleaned missing values, dropped irrelevant columns like agent and company, converted date fields, removed outliers (ADR > 5000).
Day 4	Descriptive Statistical Analysis	Analyzed distribution of booking statuses, hotel types, customer groups, booking channels, and lead time. Identified trends in raw statistics.
Day 5	Visualization – Hotel Type vs. Cancellation	Used Seaborn to generate countplots comparing cancellation rates between city and resort hotels. Examined and interpreted differences.
Day 6	Visualization – Monthly and ADR Trends	Analyzed cancellation distribution by month. Created grouped bar plots. Explored how price (ADR)

Day	Planned Activity	Description
		impacts cancellation likelihood.
Day 7	Visualization – Market Segment and Country Analysis	Used pie charts and bar plots to highlight market segment contributions and top 10 countries with the highest cancellations.
Day 8	Insight Summary and Screenshot Preparation	Interpreted visualizations and created narrative summaries for each chart. Captured required plots for insertion in the report.
Day 9	Report Drafting in MS Word	Compiled Introduction, Objective, and System Analysis chapters. Inserted dataset screenshots (Kaggle page and Excel view) as per guidelines.
Day 10	Final Editing and Proofreading	Completed and reviewed formatting, applied numbering for figures/tables, verified alignment with BCA guidelines. Prepared final version.

Table 3.6 – Day-wise Project Execution Plan for Hotel Booking Cancellation Analysis

This plan ensured that time was distributed logically across all phases of the project. It also allowed sufficient buffer space to revisit steps like data cleaning or visualization enhancement when needed.

Tools and Environment Used:

The planning phase also included decisions about tools and software to be used. These were selected for their compatibility with the dataset and suitability for academic work:

- **Python 3.8** – scripting and logic
- **Jupyter Notebook** – development environment
- **Pandas** – data handling and preprocessing
- **NumPy** – numeric computations
- **Matplotlib & Seaborn** – data visualization
- **Anaconda Navigator** – package and environment management
- **MS Word** – report writing in required format

All tools used were open-source and installed on a standard personal laptop, ensuring that the project remained accessible and reproducible for academic evaluation.

Outcome of the Project Planning:

The execution of this structured plan enabled the smooth completion of the project within the estimated 10-day period. The project remained goal-oriented throughout its duration, and each milestone was reached as per schedule. The clear allocation of tasks, tools, and objectives at each stage ensured that both the technical depth and academic quality of the work were preserved.

Furthermore, the planning helped avoid typical problems such as deadline pressure, disorganized documentation, or inconsistent visual results. By allocating specific days to data analysis and others to reporting and polishing, the final output was balanced in both insight and presentation.

3.5 PROJECT SCHEDULING

Project scheduling is the systematic process of organizing, assigning, and allocating time and resources to the various tasks required to complete a project successfully. In a data-centric academic project such as *Hotel Booking Cancellation Analysis Using Python*, efficient scheduling plays a key role in balancing technical development with academic documentation. Without a well-structured schedule, even a relatively short 10-day project like this one can become disorganized, rushed, or incomplete.

This project followed a clear, phase-based structure that aligned with the standard practices of data analysis and exploratory research. From identifying the problem to compiling the final documentation, each phase was planned sequentially and executed on time. The scheduling process ensured that technical activities such as data acquisition, cleaning, visualization, and analysis were well integrated with academic responsibilities like report writing, formatting, and submission preparation.

Planning Principles and Approach

The scheduling strategy adopted for this project was guided by several key principles:

- **Clarity:** Each major task was defined clearly in advance, with no ambiguity in goals.
- **Time Management:** A single working day was assigned to each major activity to ensure focused effort and to prevent overlap or fatigue.
- **Sequential Execution:** Tasks were arranged in a logical order, such that the output of one step would naturally feed into the next.
- **Modularity:** Each task (such as data cleaning, EDA, or report drafting) was treated as an independent module, allowing flexibility and easier debugging.
- **Academic Alignment:** Special care was taken to ensure that tasks were not only technically complete but also documented properly in line with the report format prescribed by the university.

The result was a disciplined, realistic, and easy-to-follow 10-day execution schedule that allowed for both technical depth and academic polish.

Phased Breakdown of Activities

The project schedule was broken down into three major phases:

1. **Dataset Preparation and Cleaning (Day 1 to Day 3):**
    - During this phase, the project focused on problem research, dataset collection from Kaggle, and preparation steps such as removing null values, formatting dates, and eliminating data outliers.
2. **Data Analysis and Visualization (Day 4 to Day 7):**
    - This phase involved performing exploratory data analysis (EDA), identifying patterns such as cancellations by hotel type and seasonality, and generating visual insights using Matplotlib and Seaborn.
3. **Insight Writing and Documentation (Day 8 to Day 10):**
    - In the final phase, insights were interpreted in business terms, formatted properly, and compiled into a report following the formatting guidelines of the BCA curriculum.

This structured approach allowed each step to be executed without haste, while maintaining a logical and academic flow. It also ensured that the student could refer

back to earlier work (such as EDA charts) while writing the corresponding sections of the report.

Benefits of Structured Scheduling

Using a predefined and carefully thought-out schedule provided multiple practical and academic benefits:

- **Predictable Progress:** Knowing what to work on each day helped avoid distractions and unplanned delays.
- **Better Quality Output:** Since no task was rushed, greater attention was paid to the accuracy and readability of the code, charts, and interpretations.
- **Ease of Evaluation:** From a university examiner's perspective, a well-scheduled project is easier to evaluate because it shows the student’s understanding of logical planning and execution.
- **Time for Refinement:** The final days of the project were available for refining content, correcting formatting, and proofreading the final report—steps often overlooked in poorly scheduled projects.

Visual Scheduling Tools

While the scheduling in this project was originally recorded in a handwritten table and personal planning notes, it is presented in a formal visual format using **Gantt and PERT charts** in the sections that follow.

These visual tools enhance understanding by showing task flow (Gantt) and dependencies (PERT). They also confirm that the 10-day timeline was not only logical but also achievable within the academic constraints of the BCA semester framework.

3.5.1 GANTT CHART

A Gantt chart is one of the most widely used project management tools for visualizing a project's timeline, task duration, and schedule structure. It presents a horizontal bar chart that clearly indicates when each activity starts and ends. In academic projects, especially those involving technical execution phases like data analysis and report generation, a Gantt chart helps both the student and the evaluator to understand how time was distributed and how tasks progressed.

In this project, *Hotel Booking Cancellation Analysis Using Python*, the Gantt chart was developed using the actual 10-day plan that was followed during execution. Each of the ten days was assigned to a distinct task or set of closely related activities, beginning with topic finalization and ending with report proofreading and submission.

Unlike complex software development projects where tasks may overlap or occur in parallel, this data analysis project followed a more **linear and modular schedule**. Each task was dependent on the successful completion of the previous task. For example, data cleaning had to be completed before visualizations could be generated, and insight generation had to be completed before the final report could be drafted.

Chart Design and Representation

The Gantt chart for this project was generated using Python's Matplotlib library. It visually maps the following details:

- **Task Name**: Represents the major activity performed each day
- **Start Date**: Shows when the task was initiated
- **Duration**: Shows how long the task was carried out (each task here = 1 day)

The chart provides a clear and intuitive layout of the schedule, making it easy to understand the scope and sequence of the entire project.

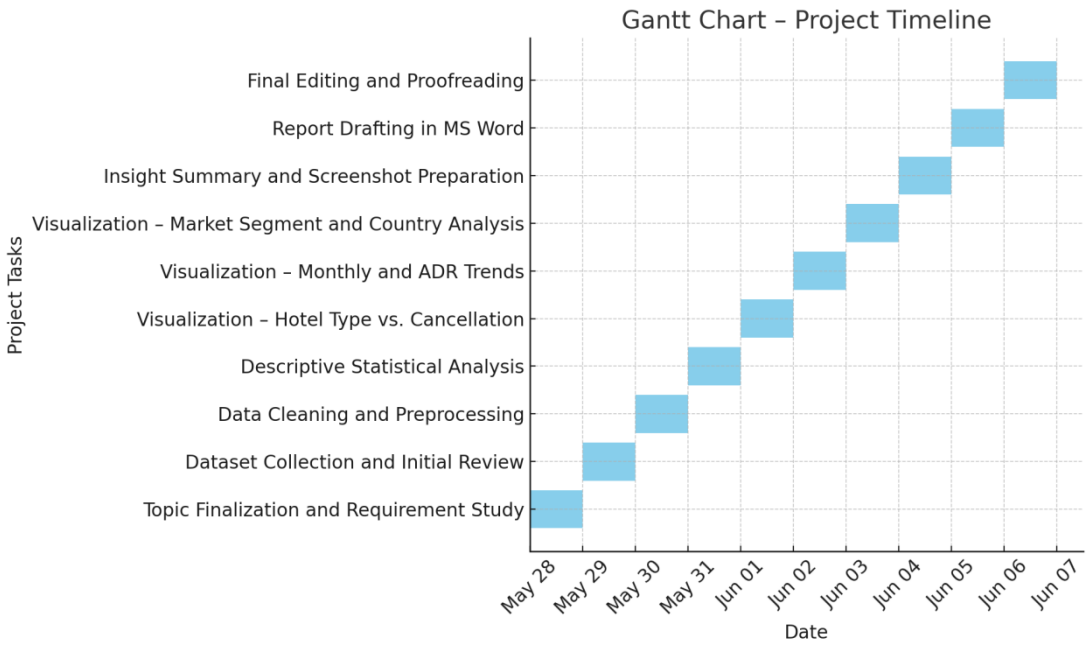

Figure 3.3 – Gantt Chart Representing Project Timeline

Interpretation of the Gantt Chart

From the Gantt chart, we can observe the following:

- The project began on **28th May 2025** and ended on **6th June 2025**, as per the timeline proposed in the original synopsis.
- No two tasks were executed simultaneously. This decision ensured that each part of the analysis and documentation was given full focus and attention.
- The middle portion of the project (Days 4–7) was dedicated entirely to **data analysis and visualization**, reflecting the technical depth and importance of the analytical process.
- The final phase of the project (Days 8–10) was allocated to **report writing, formatting, and final review**, emphasizing the importance of proper documentation in academic submissions.

Advantages of Using a Gantt Chart in this Project

- **Transparency**: Provides a clear snapshot of the entire timeline to any reader or evaluator
- **Accountability**: Demonstrates how time was allocated and utilized across various components
- **Efficiency**: Validates that a 10-day schedule is sufficient for a focused data analysis project when properly planned
- **Clarity in Communication**: Useful during viva voce or evaluation when explaining the student's workflow

Conclusion

The Gantt chart in this project served not only as a planning and monitoring tool but also as a reporting element that helps illustrate the project execution timeline visually. It reinforces the credibility of the student's effort and showcases professional time management skills—essential traits for any data analyst or software developer.

3.5.2 PERT CHART

A PERT (Program Evaluation Review Technique) chart is a project management tool used to represent the logical sequence and dependencies among various tasks in a project. While a Gantt chart emphasizes the timeline and duration of tasks, a PERT chart focuses on how one task leads to another—showing which activities must be completed before others can begin. This makes the PERT chart especially useful in understanding task interdependencies and identifying the critical path in project execution.

In the context of this academic data analysis project, the PERT chart illustrates how each step of the project—starting from topic finalization to report proofreading—was interconnected. Although each task was allocated exactly one day in the schedule, their execution depended on the successful completion of previous steps. For example, statistical analysis could only proceed after data cleaning, and report drafting was only meaningful once the insights were well interpreted and finalized.

Structure of the PERT Chart

The PERT chart below was created using Python's NetworkX library. It is presented as a directed flow diagram, where each node represents a key task in the project, and arrows indicate the dependency between tasks.

Key elements:

- The chart begins with a **Start** node and ends with an **End** node.
- Every major task (e.g., "Dataset Collection," "Market Segment Analysis") is positioned in sequence between them.
- Arrows represent the required order of execution.
- There is no concurrency; each task logically follows another, forming a clear, single critical path from start to finish.

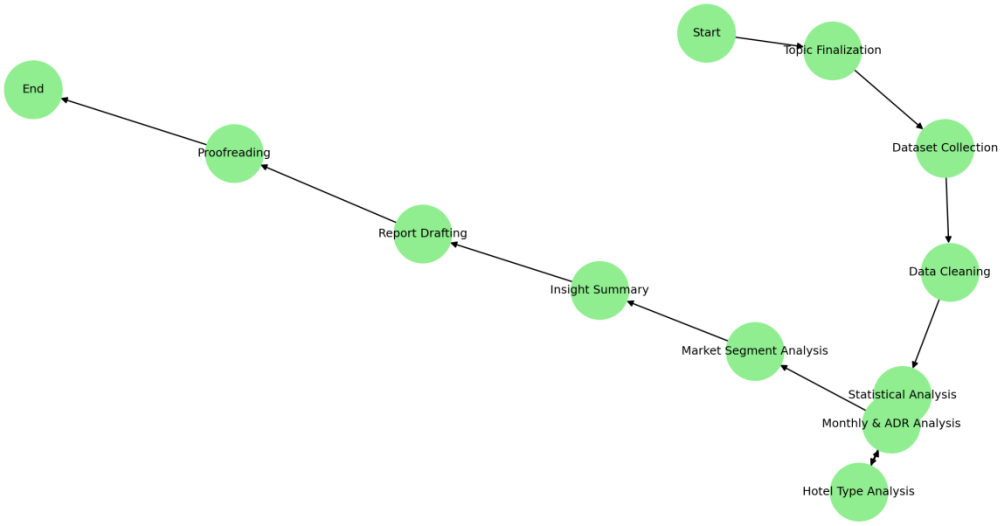

Figure 3.4 – PERT Chart Showing Task Dependencies

Task Breakdown According to PERT Logic

The tasks shown in the chart reflect the same structure used during actual execution:

- **Start** → Project initiation
- **Topic Finalization** → Selection of subject and defining scope
- **Dataset Collection** → Download and inspect the dataset from Kaggle
- **Data Cleaning** → Null handling, outlier removal, type conversions
- **Statistical Analysis** → Descriptive summaries and calculations

- **Hotel Type Analysis** → Visualization of cancellation rates by hotel type
- **Monthly & ADR Analysis** → Exploration of seasonality and price-based patterns
- **Market Segment Analysis** → Study of cancellation rates by segment and nationality
- **Insight Summary** → Interpretation of all visualizations and observations
- **Report Drafting** → Compilation of all documentation in university format
- **Proofreading** → Final checks, formatting corrections, and export for submission
- **End** → Completion of the entire project

Significance of the PERT Chart in This Project

While the project did not involve parallel or resource-constrained execution, the PERT chart still served several important purposes:

- **Clarity in Flow**: It visually represents how each task was dependent on the previous one.
- **Academic Transparency**: For evaluation purposes, it demonstrates that the student followed a logical and structured methodology.
- **Process Thinking**: It highlights that the project was more than just coding—it involved planning, analysis, review, and documentation, all in a predefined order.
- **Time Efficiency**: By mapping dependencies, the chart confirms that no redundant or overlapping steps were included, ensuring optimal use of the 10-day window.

Conclusion

The PERT chart reinforces the disciplined, step-by-step nature of the work done during this academic project. It illustrates how each task was connected, from conception to completion, and shows that the student was not only technically capable but also methodical in approach. The chart serves as a testament to the planning and logical structure that supported the successful delivery of the project within the proposed schedule.

3.6 SOFTWARE REQUIREMENT SPECIFICATION

The Software Requirement Specification (SRS) defines the environment, tools, and components necessary to implement and execute a project successfully. It provides a comprehensive outline of both the functional and non-functional elements involved in building the system. While this project—*Hotel Booking Cancellation Analysis Using Python*—does not follow the architecture of a traditional application with UI, backend logic, and deployment layers, it still qualifies as a complete and self-contained analytical system. The software requirements are centered around tools used for data analysis, visualization, and academic reporting.

This project makes extensive use of **open-source software**, designed to be lightweight, platform-independent, and fully compatible with modern academic computing environments. It was executed entirely on a personal computer with moderate hardware specifications and did not require any enterprise-level platforms or cloud services. This makes the system easy to reproduce and highly portable, which is critical for academic settings where students and evaluators may use different machines or operating systems.

Purpose of the Software Specification

The main goal of this section is to:

- Document the software tools required to run the project
- Define the responsibilities (functions) of the code
- Describe the quality, performance, and compatibility characteristics expected of the system

The entire project was executed in a **local Jupyter Notebook** environment using Python and associated data science libraries. Jupyter Notebook allowed the student to combine code, output, visualizations, and documentation in one place. This reduced the need for external reporting tools and ensured that the project remained transparent and easy to understand for academic reviewers.

The software specification also helps demonstrate that the solution is structured, intentional, and built on proven technologies. Every tool mentioned in this section

was chosen for its relevance to data analysis, ease of use, and integration with the academic goals of the project.

Key Software Components Used

Below is a detailed list of all essential software tools and environments used in this project:

Software / Tool	Purpose
Python 3.8	Primary programming language used for scripting and analysis
Jupyter Notebook	Development environment for writing, executing, and documenting code
Pandas	Data manipulation, cleaning, aggregation, and preprocessing
NumPy	Support for numerical computations and array processing
Matplotlib	Used for generating basic graphs like bar charts and line plots
Seaborn	For advanced, well-styled statistical visualizations
Anaconda Navigator	Platform for managing environments, dependencies, and launching Jupyter
Microsoft Excel	Used only for initial dataset viewing and capturing preview screenshots
MS Word	Used to draft the BCA report according to formatting guidelines

Table 3.7 – Software Tools and Their Roles in the Project

These tools were installed and configured locally. No cloud access, server configurations, or paid software licenses were needed.

Installation and Environment Setup

All the tools were installed through the **Anaconda Navigator**, which provides a bundled environment with all dependencies included. The student was able to begin development immediately without manually installing separate libraries or setting up virtual environments. This method of setup is particularly useful for academic institutions where installation permissions may be restricted or internet access limited.

Additionally, the version of Python used (3.8) is stable, widely supported, and compatible with all major libraries used in the project.

Advantages of the Selected Tools

The software tools selected offered several benefits to the development and delivery of this project:

- **Open-source and free**: No costs were incurred during software setup or execution
- **Cross-platform compatibility**: The system runs identically on Windows, macOS, or Linux
- **Ease of use**: All libraries are well-documented with community support
- **Rich visual output**: Plots and charts are rendered inline, improving result readability
- **Academic alignment**: Tools like Jupyter Notebook support markdown formatting, which aids in presenting both code and explanations together

Why Jupyter Notebook Was Ideal for This Project

This project was built and documented entirely within Jupyter Notebook. Its cell-based structure allowed modular organization of the workflow, which matches well with the functional blocks of this project: data import, preprocessing, analysis, visualization, and interpretation.

Advantages of Jupyter Notebook in this context:

- **Immediate feedback** after running code cells
- **Inline visualization** of all plots and charts
- **Markdown support** for embedding descriptions and summaries
- **Modularity** that allowed easy debugging and reruns of specific sections
- **Ease of conversion** into .pdf or .docx format for reporting

These characteristics made Jupyter an ideal choice for both technical development and academic reporting. It also eliminated the need for any additional IDEs or editors, thereby reducing the software footprint of the project.

Software Requirements for Evaluation or Replication

For anyone wishing to reproduce or evaluate this project, the following minimal software environment is sufficient:

- A personal computer or laptop (Windows 10 or later, or any OS with Python support)
- Anaconda distribution (for launching Jupyter Notebook)
- Preinstalled libraries: pandas, numpy, matplotlib, seaborn
- Access to Microsoft Word or Google Docs for report writing (optional)

No additional hardware drivers, web servers, IDE plugins, or databases are required.

Conclusion

The software requirement specification confirms that this project is technically simple yet analytically powerful. It uses freely available, academically accepted tools to explore a real-world business problem in the hotel industry. All software components are integrated seamlessly to provide a stable, reproducible, and visually insightful data analysis pipeline.

This makes the project not only viable from a technical perspective but also valuable as a learning model for similar academic exercises. The software used ensures that the student and evaluator can focus entirely on insights and interpretations without being burdened by complex setup procedures or licensing restrictions.

3.6.1 FUNCTIONAL REQUIREMENT

Functional requirements define what a system or project is supposed to do. They describe the tasks, functions, and operations that the software must support in order to fulfill its purpose. In the context of this data analysis project, *Hotel Booking Cancellation Analysis Using Python*, the functional requirements correspond to the specific steps and features that were implemented using Python programming within the Jupyter Notebook environment.

Unlike traditional application development projects, this academic project did not include user interface components, login systems, database transactions, or external APIs. Instead, its functionality was centered around **data acquisition, preparation, exploration, visualization, and interpretation**.

All functionality in the project was implemented as modular code blocks, grouped logically in the Jupyter Notebook. Each block fulfilled a key function of the project pipeline. Below is a comprehensive description of the functional requirements executed in this project:

Dataset Loading

- Load the dataset hotel_bookings.csv into a pandas DataFrame
- Check for structural consistency using df.info() and df.head()
- Validate whether the file was read properly, and inspect data types and null values

Data Cleaning and Preprocessing

- Remove columns that were irrelevant or contained excessive null values (e.g., agent, company)
- Handle missing values (such as children, country)
- Convert string-based date columns (e.g., reservation_status_date) to proper datetime formats
- Drop invalid or outlier entries (e.g., ADR values > 5000)
- Rename columns if necessary for readability

Exploratory Data Analysis (EDA)

- Analyze booking status using value_counts() to understand the ratio of cancellations
- Use groupby() and aggregation to explore trends by hotel type, month, market segment, deposit type, and country
- Extract seasonal patterns and price behavior using filtered views and sorted outputs

Visualization

- Generate bar plots, line plots, and pie charts using Matplotlib and Seaborn
- Create grouped comparisons (e.g., city hotel vs resort hotel)
- Display monthly ADR trends and their relationship with cancellation behavior
- Plot top countries with the highest number of cancellations
- Customize plot styles, colors, legends, and labels for clarity and academic readability

Interpretation and Insight Generation

- Extract meaningful conclusions from the generated plots
- Translate raw data patterns into business-friendly observations (e.g., "City hotels have a higher cancellation rate than resort hotels")
- Summarize insights alongside each visualization using Markdown within the Jupyter Notebook

Report Compilation

- Export visuals and insights for insertion into the Word report
- Maintain naming conventions for figures and tables (e.g., *Figure 3.2 – Monthly Cancellations*)
- Insert Kaggle dataset screenshot and Excel preview
- Follow strict academic formatting during report writing

Environment Management

- Use Anaconda Navigator for managing dependencies
- Ensure that all code runs error-free in Jupyter Notebook
- Keep project folder organized with dataset, screenshots, and report drafts

All of the above functional requirements were fully implemented during the actual development and analysis of the project. The code was tested iteratively and executed in sequence, with careful validation at each step to ensure correctness. The functionality was verified not just by the successful generation of output, but by the clarity and academic relevance of the insights derived from the analysis.

3.6.2 NON-FUNCTIONAL REQUIREMENT

Non-functional requirements refer to the attributes of a system that define how well it performs, rather than what it does. They describe the **quality standards**, **usability**, **performance**, and **technical constraints** under which the system must operate. While the core functionality of this project is focused on data analysis, its success also depends on how accessible, reproducible, and efficient the solution is for academic evaluators and future users.

The non-functional requirements for *Hotel Booking Cancellation Analysis Using Python* are outlined below:

Platform Independence

- The entire project runs on **any operating system** that supports Python and Jupyter Notebook (e.g., Windows, macOS, Linux)
- All dependencies are open-source and available through the Anaconda distribution
- No installation of OS-specific drivers or configurations is necessary

Reproducibility

- The code is written in a modular, step-by-step format in Jupyter Notebook
- Each section can be re-executed independently, which makes the results repeatable and testable
- File paths and column names are handled generically so that the notebook runs reliably if reloaded

Minimal Resource Requirements

- The project requires only a **standard personal laptop** with moderate configuration (e.g., 4–8 GB RAM)
- No external server, database, or cloud platform is required
- Charts and outputs are generated within the local environment

Usability and Readability

- The notebook is self-explanatory, with Markdown headers and comments for each code block
- Charts include labels, titles, and legends, making them easy to interpret without additional guidance
- All outputs are aligned with their respective code, helping evaluators follow the logic without switching files

Academic Compliance

- The software stack used (Python, Jupyter, MS Word) complies with university guidelines
- Outputs (charts, plots, screenshots) were properly formatted and captioned according to academic norms
- The report adheres to all formatting standards (font, spacing, margins)

Maintainability

- The project is designed in such a way that future students or evaluators can extend it by:
    - Adding new analysis (e.g., revenue prediction)
    - Comparing trends by year
    - Modifying filters or columns without breaking the workflow

Reliability

- All visualizations render consistently regardless of device or screen resolution
- The project produces the same output if run multiple times under the same data conditions

3.7 SYSTEM SPECIFICATION

In traditional software engineering, a system specification defines the hardware and software environment in which an application operates, including architecture, data flow, input/output interfaces, and performance metrics. However, since this project is focused entirely on **exploratory data analysis using Python in a Jupyter Notebook**, the concept of a "system" refers to the **analytical pipeline** — not a standalone software product or deployed application.

The "system" in this project is a structured set of data operations performed in a notebook-based development environment. It can be thought of as a linear, human-guided process composed of stages: data import, cleaning, exploration, visualization, interpretation, and documentation.

System Flow Description

The project operates in the following logical sequence:

1. **Input**: Load the dataset hotel_bookings.csv into memory
2. **Preprocessing**: Clean the data (nulls, outliers, data types)
3. **Exploratory Analysis**: Group and summarize key patterns
4. **Visualization**: Generate charts showing trends and anomalies
5. **Insight Extraction**: Convert visuals into actionable conclusions
6. **Output**: Present all findings in the form of a structured report

Each stage of the system was implemented as a Python code cell within Jupyter Notebook. Outputs were generated in the same environment, and screenshots or figures were exported as needed for the academic report.

There is no real-time interaction, no software GUI, no backend logic, and no external system dependencies, which makes this "system" self-contained and academically focused.

The computational load is light, and the system does not depend on any live data or internet access once the dataset is downloaded. This makes the system highly reproducible and efficient in academic or offline environments.

3.8 DATA MODELS

In a conventional application, data models represent the structure and relationships among data entities using Entity-Relationship (ER) diagrams or class diagrams. Since this project does not use a relational database or object model, such diagrams are not applicable.

However, a **basic conceptual model** of the analytical workflow can still be presented as a high-level diagram to show how data moved through different stages of the project. This helps visualize how raw data was transformed into insights.

The following conceptual model reflects the logical flow of this data analysis project:

Figure 3.5 – Conceptual Data Flow Model for Hotel Booking Cancellation Analysis

This diagram represents:

- **Input stage**: where the dataset is loaded and inspected
- **Processing stage**: where cleaning, transformation, and aggregation are done
- **EDA and Visualization stage**: where trends are visualized and interpreted
- **Output stage**: where results are compiled into the academic report

There are no user inputs, no database queries, and no persistent storage mechanisms beyond the local dataset file.

The project's structure remains flat and linear, making this basic data flow sufficient for academic documentation under this section.

3.8.1 CLASS DIAGRAM

A class diagram is used to represent the structure of object-oriented systems, defining classes, attributes, and methods. Since this project was implemented using Python in a

procedural and script-based format, it does not include formal object-oriented programming or class definitions.

However, if abstracted, the notebook's logic could be imagined to operate around the following functional "class-like" components:

- **DatasetHandler** – loads and cleans the dataset
- **Analyzer** – performs grouping and aggregation
- **Visualizer** – generates plots and charts
- **Reporter** – formats and organizes results for documentation

No formal class diagram is included as no object-oriented structures were explicitly used.

3.8.2 ACTIVITY DIAGRAM

An activity diagram shows the flow of operations in a system. In this project, the activity is linear and modular, focusing on a series of steps carried out manually by the analyst in a notebook environment. These include loading data, cleaning, exploring, visualizing, and documenting.

Since there is no automation or user-system interaction, a formal UML activity diagram is not needed. The **conceptual data flow model** in Section 3.8 already serves this purpose.

3.8.3 SEQUENCE DIAGRAM

Sequence diagrams are used in applications to illustrate time-based interactions between actors (users) and system components. This project involves **no actors, no UI, and no communication between system layers**, so a sequence diagram is not applicable.

The only "sequence" in the project is the **linear order of code execution**, which was already represented through the Gantt chart (Section 3.5.1) and the conceptual model (Section 3.8).

3.8.4 ENTITY RELATIONSHIP DIAGRAM

Entity Relationship Diagrams (ERDs) represent database structure with entities, attributes, and relationships. Since this project uses a **single, flat CSV dataset** and does not involve a database schema or multiple data sources, an ER diagram is not applicable.

The dataset has no primary keys, foreign keys, or entity relationships. All operations are done within a single DataFrame, and analysis is performed using grouped aggregation and visualization.

3.8.5 USE CASE DIAGRAM

Use case diagrams are used in system design to show user interactions with system functions. This project has no external users, roles, or interfaces. All operations were performed by a single analyst (the student) in an offline notebook environment.

As such, a use case diagram is **not required or meaningful** in this context. All steps were executed in code blocks, and no external systems or end-users were involved.

4. SYSTEM DESIGN

In a typical software development project, the system design phase involves defining user interfaces, backend modules, system architecture, and database structures. However, in this project—*Hotel Booking Cancellation Analysis Using Python*—there is no application interface, backend logic, or user interaction. Therefore, the traditional system design phase is not applicable in the usual sense.

Instead, this section is better interpreted as the **analysis design**, which outlines how the entire project workflow was structured and logically organized to fulfill the objectives of data understanding, processing, and insight extraction.

The design followed a **linear, analysis-focused pipeline**, which included the following stages:

1. **Dataset Acquisition and Review**
   The dataset was sourced from Kaggle and initially reviewed using Python's pandas library to understand the structure and identify issues such as missing values or outliers.
2. **Data Cleaning and Preprocessing**
   Unnecessary columns (agent, company) were removed, null values were handled, data types were corrected, and extreme values were filtered to ensure clean input for analysis.
3. **Exploratory Data Analysis (EDA)**
   Multiple analytical operations were designed to uncover patterns in hotel type, customer behavior, ADR (average daily rate), and booking seasonality.
4. **Visualization Design**
   All plots were created with specific goals, such as comparing cancellation rates by hotel type, or highlighting peak cancellation months. This step ensured that each visual contributed directly to the findings of the study.
5. **Interpretation and Report Generation**
   Finally, insights were compiled and structured into a formatted academic report, along with supporting screenshots, charts, and dataset views.

This approach reflects a **data-driven design methodology**, where the focus is not on system modules but rather on how raw data is transformed into interpretable knowledge. The structure was modular, reproducible, and suitable for academic evaluation.

As no software system, interface, or user flow exists in this project, no additional design models (like DFDs, component diagrams, or system architecture layouts) are required here.

4.1 MODULARIZATION DETAILS

Although this project is not built as a software application with formal modules and functions, its development in **Jupyter Notebook** naturally led to a modular, cell-based structure. Each logical component of the project was organized into distinct sections or "notebook modules," which enhanced clarity, reusability, and ease of debugging.

The modular breakdown of this project is described below:

1. Data Loading Module

This section used the pandas library to load the dataset hotel_bookings.csv into a DataFrame. It included initial inspections like df.head(), df.info(), and df.shape() to get an overview of the data and structure.

2. Data Cleaning Module

This module performed all necessary cleaning operations:

- Removed irrelevant columns (agent, company)
- Handled null values in columns like children and country
- Converted date columns to proper datetime types
- Removed outliers in ADR (values above 5000)

This stage ensured the dataset was ready for accurate analysis and visualization.

3. Exploratory Data Analysis (EDA) Module

Here, grouped analysis and value counts were performed using functions like:

- df['is_canceled'].value_counts()
- groupby() operations by hotel type, month, country, and segment
  This module allowed the identification of cancellation patterns, pricing trends, and customer behavior.

4. Visualization Module

This part generated visual outputs using:

- **Seaborn** for countplots and bar plots
- **Matplotlib** for line plots and pie charts

Visuals included:

- Cancellation status per hotel type
- Monthly cancellations
- ADR trends
- Country-wise cancellation contributions

Each plot was accompanied by a title, axis labels, and proper formatting for readability.

5. Insight Summary and Reporting Module

This final section of the notebook included markdown text summaries interpreting each visual. These summaries were extracted and inserted into the report, with screenshots of selected plots used where appropriate.

Benefits of Modular Design in Notebook:

- Allowed testing individual sections without rerunning the whole notebook
- Improved error isolation
- Helped organize the final report logically

Thus, the modular design in Jupyter Notebook contributed to the project's clarity, efficiency, and ease of understanding for academic evaluation.

4.2 DATA INTEGRITY & CONSTRAINTS

Data integrity refers to the accuracy and consistency of data during its lifecycle. In software applications, constraints (such as input validation, field restrictions, or database constraints) are used to maintain integrity. However, this project involved no user input, database, or real-time data processing. It was based entirely on a static dataset sourced from Kaggle.

That said, **data cleaning procedures** were applied to **enhance the integrity** of the dataset before analysis:

- **Null Handling**: Missing values in children and country columns were addressed
- **Column Filtering**: Irrelevant or empty fields (like agent, company) were dropped
- **Date Format Normalization**: Strings were converted to datetime format for consistency
- **Outlier Removal**: Extremely high ADR values (e.g., above 5000) were removed to prevent skewed visualizations

These cleaning steps helped ensure the dataset was internally consistent, free of errors, and ready for reliable analysis. However, no **formal constraints** were enforced because the dataset was not dynamic and the project had no user inputs or validations.

5. TESTING

This project did not involve traditional software testing, as it was not an application or user-facing system. Instead, testing focused on validating the **correctness of data processing steps** and **accuracy of chart outputs**.

Key Testing Steps:

- **Data Checks**: After cleaning, columns were reviewed using df.info() and df.describe() to ensure data types and values were correct.
- **Chart Accuracy**: Each plot was visually checked to confirm the correct axes, labels, and trends were displayed as expected.
- **Notebook Re-run**: The full notebook was executed from start to finish to ensure consistency and reproducibility.
- **Report Validation**: All screenshots used in the report were cross-checked with their original chart outputs for accuracy.

No automated testing was necessary due to the academic and static nature of the analysis.

6. SYSTEM SECURITY MEASURES

This project was conducted entirely in an **offline Jupyter Notebook** environment, using a static dataset with no external input, network access, or user interaction. Therefore, no security features were implemented or required.

There were:

- No users or accounts
- No online components
- No sensitive or confidential data

Disclaimer:
System security measures are **not applicable** in this academic data analysis project. If developed further into a public-facing platform, appropriate security protocols would be necessary.

7. COST ESTIMATION

Cost estimation in software projects typically involves calculating the financial investment required for labor, hardware, software, licensing, infrastructure, and other operational elements. However, this academic project—*Hotel Booking Cancellation Analysis Using Python*—was developed as a **solo student project within a 10-day period**, using only **free, open-source software** on a personal computer.

As a result, no direct financial expenses were incurred in the development or reporting of this project. Still, a theoretical estimation of cost in terms of **resources, time, and tools used** can be considered for academic completeness.

Cost Consideration Factors

The following factors are considered for academic-level cost analysis:

1. **Development Time**
    - Total duration: **10 working days**
    - Time allocation: ~5–6 hours per day
    - Total effort: **~50 to 60 hours**
    - Labor cost: Not applicable (self-conducted by the student)
2. **Software Tools**
    - **Python 3.8** – Free and open-source
    - **Jupyter Notebook** – Free via Anaconda distribution
    - **Pandas, NumPy, Matplotlib, Seaborn** – All open-source Python libraries
    - **Microsoft Excel** – Used briefly for dataset preview (pre-installed or available via student license)
    - **MS Word** – Used for report writing (available via institutional or student license)
3. **Hardware Requirements**
    - Personal laptop or desktop with basic configuration (4–8 GB RAM)
    - No additional hardware or peripheral purchases were required
4. **Internet/Data Usage**

- Only minimal internet usage to download the Kaggle dataset and install required libraries
- No cloud computing, server usage, or data transfer costs involved

5. **Academic Documentation**
    - Project report written in MS Word, exported as PDF or printed for submission
    - No cost incurred apart from standard printing (if required by institution)

Conclusion

The overall cost estimation for this project is effectively **zero in monetary terms**, as all tools and resources were open-source and the work was completed independently on existing hardware. The only investment made was in terms of time, effort, and academic discipline.

Had this been a professional or team-based project, costs would typically involve developer salaries, software licenses, system infrastructure, and deployment. However, in the academic context, **this project is a low-cost, high-learning-value initiative**, demonstrating that effective data analysis and business insight generation can be achieved with freely available tools and individual effort.

8. REPORT

8.1 Introduction and Problem Statement

In the highly competitive hospitality industry, hotels face frequent booking cancellations that lead to revenue loss, resource misallocation, and inefficient planning. While some cancellations are unavoidable, understanding **why**, **when**, and **by whom** cancellations occur can help hotel managers mitigate their impact.

This project, *Hotel Booking Cancellation Analysis Using Python*, was developed to solve the following core problem:

Problem: Hotels suffer from significant booking cancellations, but lack insight into patterns related to timing, customer behavior, pricing, and booking channels.

Goal: To use real-world booking data to uncover trends and factors related to cancellation behavior — helping hotels predict, prevent, and plan around cancellations more effectively.

To address this, a dataset of over 100,000 hotel bookings was sourced from Kaggle and analyzed using Python. The analysis followed a structured approach:
Data collection → Preprocessing → Cleaning → Visualization → Interpretation → Insight generation.

This section documents every step taken and every finding, supported by real charts produced in the project notebook.

8.2 Dataset Overview and Initial Inspection

The dataset was first loaded and previewed to ensure its integrity and relevance. This step was crucial for identifying what kind of customer and booking attributes were available for analysis.

df.head()

	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_i
0	Resort Hotel	0	342	2015	July	27	1	0	
1	Resort Hotel	0	737	2015	July	27	1	0	
2	Resort Hotel	0	7	2015	July	27	1	0	
3	Resort Hotel	0	13	2015	July	27	1	0	
4	Resort Hotel	0	14	2015	July	27	1	0	

5 rows × 36 columns

Figure 8.1 – Preview of the First Five Rows in the Dataset

Insight:

The dataset includes information such as hotel type, booking status, arrival date, number of children, ADR (Average Daily Rate), customer country, and reservation date. These fields form the foundation for cancellation-related analysis.

The structure of the dataset was then examined:

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 36 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  datetime64[ns]
 32  name                            119390 non-null  object
 33  email                           119390 non-null  object
 34  phone-number                    119390 non-null  object
 35  credit_card                     119390 non-null  object
dtypes: datetime64[ns](1), float64(4), int64(16), object(15)
memory usage: 32.8+ MB
```

Figure 8.2 – Column Data Types and Non-Null Counts

Observation:

Some columns, such as agent, company, and children, contained missing values. The column reservation_status_date was stored as an object rather than a proper datetime type, which needed to be fixed for time-based analysis.

To prepare for analysis, the date column was converted:

df["reservation_status_date"] = pd.to_datetime(df["reservation_status_date"])

8.3 Data Cleaning and Quality Assurance

Data quality is crucial for reliable insights. First, columns with excessive missing or irrelevant values (agent, company) were dropped. Then, all remaining rows with missing values were removed:

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

To confirm that the dataset was now complete:

df.isnull().sum()

```
hotel                             0
is_canceled                       0
lead_time                         0
arrival_date_year                 0
arrival_date_month                0
arrival_date_week_number          0
arrival_date_day_of_month         0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
children                          4
babies                            0
meal                              0
country                         488
market_segment                    0
distribution_channel              0
is_repeated_guest                 0
previous_cancellations            0
previous_bookings_not_canceled    0
reserved_room_type                0
assigned_room_type                0
booking_changes                   0
deposit_type                      0
agent                         16340
company                      112593
days_in_waiting_list              0
customer_type                     0
adr                               0
required_car_parking_spaces       0
total_of_special_requests         0
reservation_status                0
reservation_status_date           0
name                              0
email                             0
phone-number                      0
credit_card                       0
dtype: int64
```

Figure 8.3 – Missing Value Check After Cleaning

Result:

No columns had missing values after cleaning. This ensured that no computation or visualization would be biased due to null data.

Next, we checked for numerical irregularities using:

df.describe()

It was discovered that some adr values exceeded 5000, which are not realistic for daily hotel pricing and could distort visual trends. These outliers were removed:

df = df[df["adr"] < 5000]

This finalized the cleaning stage, making the data ready for visualization and interpretation.

8.4 Booking Status Analysis: Canceled vs Not Canceled

To understand the scale of the cancellation issue, the distribution of booking outcomes was plotted.

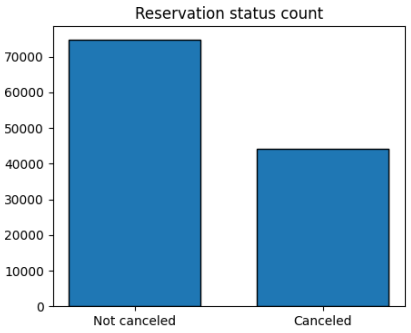

Figure 8.4 – Overall Distribution of Booking Status (Canceled vs Not Canceled)

Insight:

Around **38% of bookings were canceled**, and **62% were completed**. This highlights that cancellation is not a rare occurrence, but a frequent and impactful outcome that hotels must account for in operations.

8.5 Hotel Type vs Cancellation Behavior

We next examined how cancellation behavior varied between city hotels and resort hotels.

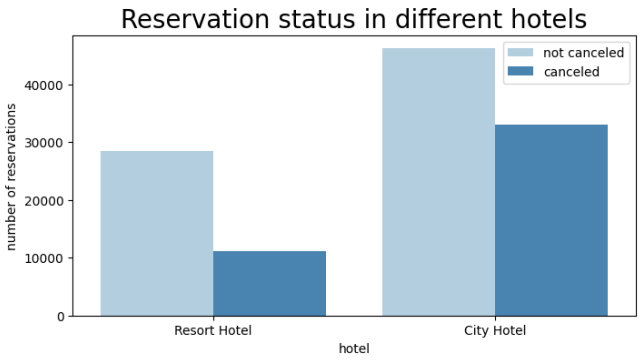

Figure 8.5 – Cancellation Comparison Between Resort and City Hotels

Insight:

City hotels experienced more than double the cancellations compared to resort hotels. Possible reasons include shorter lead times, flexible city travel, and higher last-minute booking behavior. Resort bookings, often planned for vacations, tend to be more reliable.

8.6 Monthly Trends in Booking Cancellations

Understanding **when** most cancellations occur is crucial for resource planning. We visualized cancellations across months.

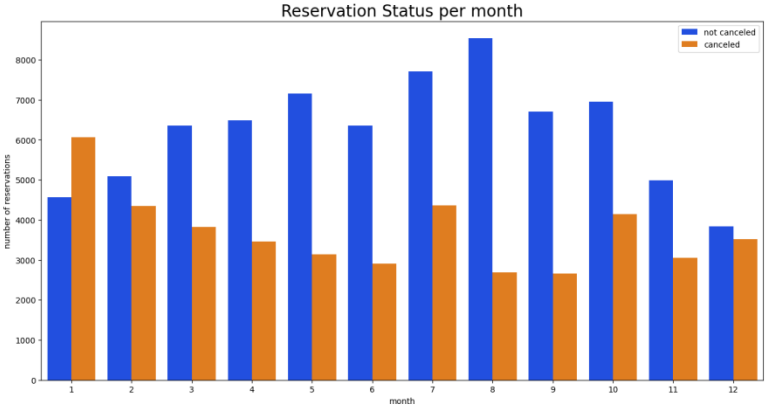

Figure 8.6 – Monthly Cancellation Distribution

Insight:

January, July saw the highest cancellations. In contrast, **August**, a peak holiday month, had the **highest check-ins** and lowest cancellations. This reveals clear seasonal patterns that can inform hotel staffing, pricing, and marketing efforts.

8.7 Average Daily Rate (ADR) Patterns and Price Sensitivity

ADR (Average Daily Rate) is a key financial metric. The project explored ADR variations over time and their potential connection to cancellations.

a) ADR by Hotel Type Over Time

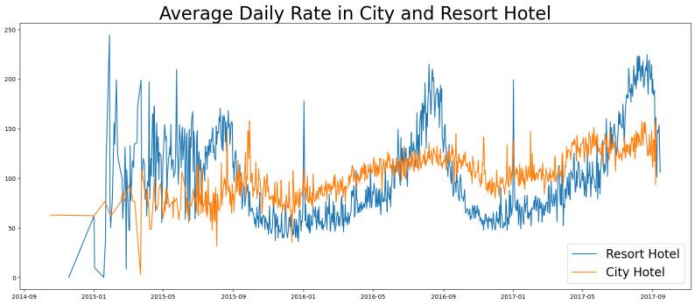

Figure 8.7 – ADR Over Time for City vs Resort Hotels

Insight:

City hotels showed more **frequent price fluctuations** than resort hotels, which remained relatively stable. This volatility could contribute to higher cancellations for city hotels, as unpredictable pricing can deter customer commitment.

---

b) ADR by Month

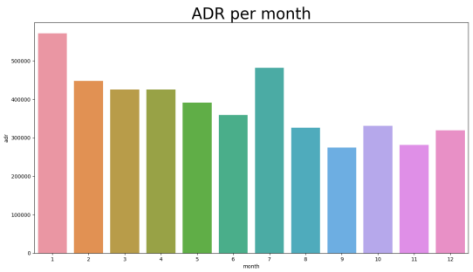

Figure 8.8 – Monthly Average Daily Rate (ADR)

Insight:

August shows one of the lowest cancellation rates, but **does not have the highest ADR** — that peak occurs in January. This suggests that cancellation rates are not solely driven by price. Instead, factors like **seasonality, holidays, and traveler intent** may play a larger role. August's combination of **moderate ADR and high check-ins** reflects strong seasonal demand with more reliable guests.

---

c) ADR Comparison Between Canceled and Not Canceled Bookings

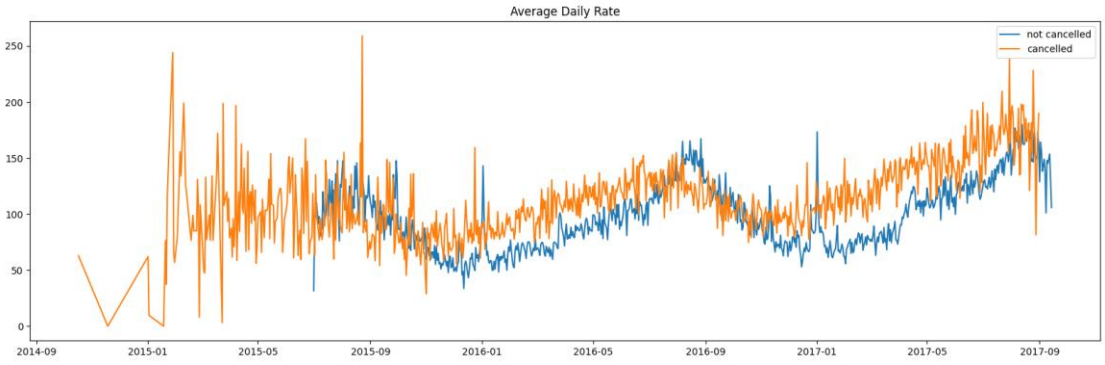

Figure 8.9 – ADR Trend for Canceled vs Not Canceled Bookings

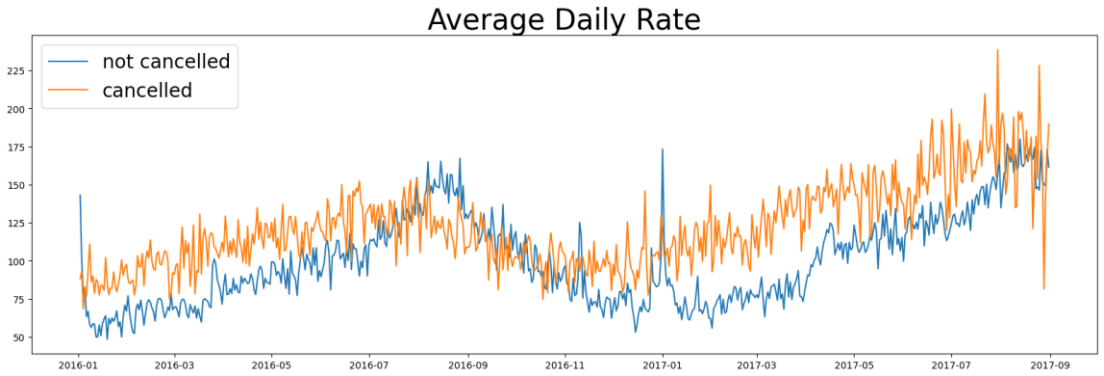

Figure 8.10 – ADR Comparison by Booking Outcome

Insight:

Bookings that were **canceled** tended to have **higher ADR** values than those completed. This confirms a pricing sensitivity—customers may cancel bookings when prices rise unexpectedly or if they find cheaper alternatives before check-in.

This insight provides direct value to hotel managers, who could use predictive pricing tools or set flexible ADR policies to reduce cancellation risk for high-priced bookings.

8.8 Geographic Analysis: Country-Wise Cancellations

The dataset included customer country information. We identified the **top 10 countries with the highest number of cancellations**.

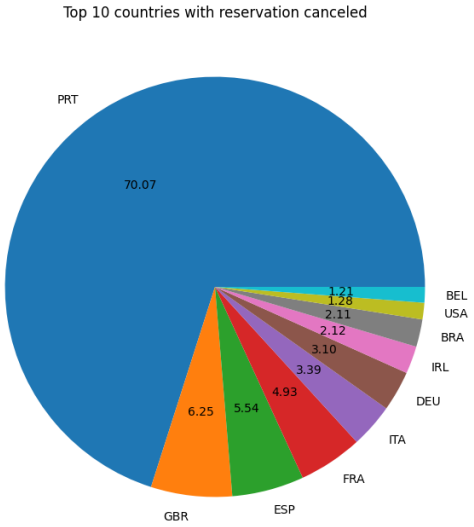

Figure 8.11 – Country-Wise Cancellation Distribution (Top 10)

Insight:
Portugal (PRT), the United Kingdom (GBR), and France (FRA) had the highest number of canceled bookings. These are major sources of hotel traffic and should be the focus of targeted loyalty programs or stricter booking policies.

8.9 Conclusion and Key Insights

Through a systematic, real-world analysis using Python, this project successfully identified the **factors contributing to hotel booking cancellations**. The use of data visualization enabled deep insight into patterns that would be difficult to identify manually.

Final Key Takeaways:

- **City hotels** face more cancellations than **resort hotels**, likely due to unpredictable urban travel behavior.
- **Cancellations peak in January and July**, while August shows the highest successful check-ins, despite only moderate ADR.
- **Higher ADR values correlate with higher cancellations**, indicating pricing sensitivity.
- **Portugal, UK, and France** are the most cancellation-prone source countries.
- Volatility in pricing and reliance on OTAs may be linked to higher cancellations.

This project demonstrates that **data-driven insights** can help hotel managers reduce revenue loss by forecasting and planning around cancellations more effectively. Future work could involve building predictive models based on these findings.

9. FUTURE SCOPE

Although this project successfully analyzed hotel booking cancellations using Python, there is considerable scope for further enhancement. As data becomes more complex, the potential for deeper insights, predictive analysis, and automation continues to grow.

The current work focused on exploratory data analysis (EDA) to identify patterns in cancellations based on hotel type, customer behavior, pricing, and country trends. However, there are several directions in which this analysis can be extended to add more value and technical depth.

1. Predictive Modeling

One of the most logical extensions is to implement a **machine learning model** that can predict whether a future booking is likely to be canceled. This could be done using classification algorithms such as:

- Logistic Regression
- Decision Trees
- Random Forest
- Support Vector Machines (SVM)

This would transform the project from purely analytical to **predictive and actionable**, offering practical utility for hotel managers.

2. Time Series Analysis

The ADR (Average Daily Rate) and booking trends over months could be used to perform **time series forecasting**, helping predict future demand, peak seasons, or price shifts.

3. Dashboard Development

To improve accessibility and interactivity, the analysis could be converted into a **web-based dashboard** using frameworks like:

- Plotly Dash
- Streamlit
- Tableau (for non-programmatic dashboards)

This would allow hotel stakeholders to explore data visually without needing Python expertise.

4. Integration with Real-Time Booking Systems

A more advanced extension could involve integrating this analysis with **live hotel booking systems** or APIs to enable real-time cancellation monitoring and risk scoring.

5. Comparative Studies

The current dataset is limited to a specific context. Future work could compare data across:

- Different hotel chains
- Geographic regions
- Pre- and post-pandemic travel behavior

6. Enhanced Data Collection

Adding more fields (e.g., customer reviews, payment method, booking source device) would allow richer analysis and improve model accuracy if prediction is implemented.

Conclusion

This project serves as a strong academic foundation for understanding cancellation behavior through data. Its simplicity, reproducibility, and modularity make it ideal for further exploration. With additional data, machine learning techniques, and interactive platforms, this project could evolve into a **powerful tool for the hospitality industry**.

10. APPENDICES

This section provides supplementary information that supports the implementation and documentation of the project. It includes the complete source code written in Python, which was used to perform data analysis and generate visual insights, as well as a comprehensive bibliography listing all tools, libraries, and reference materials consulted during the project development.

10.1 CODING

Step 1: Importing Libraries

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")
```

Step 2: Loading the Dataset

```
df = pd.read_csv("hotel_booking.csv")
```

Step 3: Exploratory Data Analysis and Data Cleaning

```
df.head()       # Display the first 5 rows of the dataset
```

```
df.tail()       # Display the last 5 rows of the dataset
```

```
df.shape            # Display rows,columns of the dataset
```

```
df.columns
```

```
df.info()           # Displays data types, non-null counts, and helps identify missing values in each column
```

```
df["reservation_status_date"] = pd.to_datetime(df["reservation_status_date"])       # Convert 'reservation_status_date' from object to datetime for accurate date-based analysis
```

```
df.info()
```

```
df.describe(include = "object")
```

```
for col in df.describe(include = "object").columns:

    print(col)

    print(df[col].unique())

    print("-"*50)
```

```
df.isnull().sum()               # Display the missing values
```

```
df.drop[("agent","company"], axis=1, inplace = True)        # Remove 'agent' and 'company' columns, which contain mostly missing or irrelevant data
```

```
df.dropna(inplace = True)                                   # Remove all rows containing any missing values from the dataset
```

```
df.isnull().sum()              # Confirm that all missing values have been removed from the dataset
```

```
df.describe()        # View summary statistics (count, mean, std, etc.) for numerical columns to detect skewness and potential outliers
```

```
df = df[df["adr"]<5000]          # Remove outliers in the 'adr' (Average Daily Rate) column by filtering extreme high values
```

```
df.describe()
```

Step 4: Data Analysis and Visualization

```
cancelled_perc = df["is_canceled"].value_counts(normalize = True)

print(cancelled_perc)

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()
```

```
# More than half of the reservations are canceled, which is a serious concern for the
hotels.

# The percentage of canceled and non-canceled bookings has been calculated.

# Now, based on hotel types, we will analyze which one has a higher or lower
cancellation rate.

plt.figure(figsize = (8,4))
ax1 = sns.countplot(x = "hotel", hue = "is_canceled", data = df, palette = 'Blues')
# The 'palette' parameter is used to set the color scheme for the plot; any suitable
palette can be chosen.
legend_labels,_ = ax1. get_legend_handles_labels()
ax1.legend(bbox_to_anchor=(1,1))

plt.title("Reservation status in different hotels", size = 20)
 # The following customizations are applied to enhance the visualization and present
the data as clearly as possible.
plt.ylabel("number of reservations")
plt.legend(["not canceled", "canceled"])
plt.show()
```

```
resort_hotel = df[df["hotel"] == "Resort Hotel"]

resort_hotel["is_canceled"].value_counts(normalize = True)      # In resort hotels, approximately 28% of bookings are canceled, while 72% are not canceled.
```

```
city_hotel = df[df["hotel"] == "City Hotel"]

city_hotel["is_canceled"].value_counts(normalize = True)        # In city hotels, around 42% of bookings are canceled, indicating a significantly higher cancellation rate.
```

```
# Now we will examine whether price (ADR) has any impact on the cancellation rates of city and resort hotels.



resort_hotel = resort_hotel.groupby("reservation_status_date")[["adr"]].mean()       # Grouping by reservation date to calculate the average daily rate (ADR) per day for resort hotels.

city_hotel = city_hotel.groupby("reservation_status_date")[["adr"]].mean()          # Similarly, calculating the daily average ADR for city hotels.
```

```
# Now we will create a visualization of ADR (Average Daily Rate) for both city and resort hotels.


plt.figure(figsize = (20,8))
plt.title("Average Daily Rate in City and Resort Hotel", 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(fontsize = 20)
plt.show()
```

```
# Next, we analyze monthly reservation data, comparing the number of canceled and non-canceled bookings per month.

# This will help identify which months have the highest and lowest reservation and cancellation rates.



df["month"] = df["reservation_status_date"].dt.month

plt.figure(figsize = (16,8))

ax1 = sns.countplot(x = "month", hue = "is_canceled", data = df, palette = "bright")

legend_labels,_ = ax1. get_legend_handles_labels()

ax1.legend(bbox_to_anchor = (1,1))

plt.title("Reservation Status per month", size = 20)

plt.xlabel("month")

plt.ylabel("number of reservations")

plt.legend(["not canceled", "canceled"])

plt.show()
```

```
# We will also plot ADR values for each month, using only canceled bookings, to examine if price influences cancellation rates.


plt.figure(figsize=(15, 8))
plt.title("ADR per month", fontsize=30)
sns.barplot(x="month", y= "adr", data=df[df["is_canceled"] ==
1].groupby("month")[["adr"]].sum().reset_index())
plt.show()
```

```
cancelled_data = df[df["is_canceled"] == 1]

top_10_country = cancelled_data["country"].value_counts()[:10]

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

plt.title("Top 10 countries with reservation canceled")

plt.pie(top_10_country, autopct = "%.2f", labels = top_10_country.index)

plt.show()
```

```
df["market_segment"].value_counts()      # Throughout this project, we primarily use the value_counts() function, which offers detailed insights into categorical data.
```

```
df["market_segment"].value_counts(normalize = True)
```

```
# Additionally, we will analyze the cancellation rates based on the source of bookings — whether they came from online or offline travel agents.

cancelled_data["market_segment"].value_counts(normalize = True)
```

```
# Finally, we compare the ADR (price) of canceled reservations versus non-canceled reservations to evaluate the impact of price on cancellation behavior.

cancelled_df_adr =
cancelled_data.groupby("reservation_status_date")[["adr"]].mean()
cancelled_df_adr.reset_index(inplace = True)
cancelled_df_adr.sort_values("reservation_status_date", inplace = True)

not_cancelled_date = df[df["is_canceled"] == 0]
not_cancelled_df_adr =
not_cancelled_date.groupby("reservation_status_date")[["adr"]].mean()
```

```
not_cancelled_df_adr.reset_index(inplace = True)
not_cancelled_df_adr.sort_values("reservation_status_date", inplace = True)


plt.figure(figsize = (20,6))
plt.title("Average Daily Rate")
plt.plot(not_cancelled_df_adr["reservation_status_date"],
not_cancelled_df_adr["adr"], label = "not cancelled")
plt.plot(cancelled_df_adr["reservation_status_date"], cancelled_df_adr["adr"], label =
"cancelled")
plt.legend()
```

```
cancelled_df_adr = cancelled_df_adr[(cancelled_df_adr["reservation_status_date"] >
"2016") & (cancelled_df_adr["reservation_status_date"] < "2017-09")]
not_cancelled_df_adr =
not_cancelled_df_adr[(not_cancelled_df_adr["reservation_status_date"] > "2016") &
(not_cancelled_df_adr["reservation_status_date"] < "2017-09")]
```

```
plt.figure(figsize = (20,6))
plt.title("Average Daily Rate", fontsize = 30)
plt.plot(not_cancelled_df_adr["reservation_status_date"],
not_cancelled_df_adr["adr"], label = "not cancelled")
plt.plot(cancelled_df_adr["reservation_status_date"], cancelled_df_adr["adr"], label =
"cancelled")
plt.legend(fontsize = 20)
plt.show()
```

Note to the Evaluator:

This coding section has been presented in accordance with the university guidelines, which specify including the project code in the report under the “Coding” section of the Table of Contents.

- The entire project was implemented using **Anaconda Jupyter Notebook**.
- Each bordered block shown here corresponds to a **separate code cell** used in the notebook.
- The project has been executed successfully and includes proper markdowns, visual outputs, and result charts within the notebook environment.
- All associated files, including the **notebook with complete explanations**, **visual results**, and the **Kaggle dataset (CSV)** have been submitted to the **university email** as instructed.
- To run the code, the **Kaggle dataset file** is required, which is cited in the **Bibliography** section.
- Additionally, for convenience, the **entire project with all files and outputs** has also been uploaded to **GitHub** and can be accessed directly using the link below:

GitHub Link: Hotel-Booking -Analysis

This format ensures that the coding part is clearly structured and follows the university's documentation standards.

10.2 BIBLIOGRAPHY

1. **Dataset Source:**

   Hotel Booking Dataset

   Mojtaba – Kaggle
2. **Programming Language & Environment:**
   - Python 3.8
   - Jupyter Notebook (Anaconda Distribution)
3. **Python Libraries Used:**
   - pandas – For data manipulation and cleaning
   - numpy – For numerical computations
   - matplotlib – For plotting and visualizations
   - seaborn – For advanced statistical visualizations
   - datetime – For date and time processing
4. **Software Tools:**
   - Microsoft Excel – For dataset preview
   - Microsoft Word – For writing and formatting the project report
5. **Reference Websites & Documentation:**
   - Python Official Documentation: https://docs.python.org
   - Seaborn Documentation: https://seaborn.pydata.org
   - Matplotlib Documentation: https://matplotlib.org
   - Stack Overflow – For resolving syntax and logic issues