### INTRODUCTION

Hotel Management is a CLI-based application that uses Python, pandas, and matplotlib to manage hotel reservations. The application uses pandas to read and manipulate data from a CSV file. matplotlib is used to plot graphs of the data, which can be used to visualize information about hotel reservations.

The application is designed to be easy to use and can be used by anyone with basic knowledge of Python. It is a valuable tool for hotels of all sizes, as it can help to automate tasks, improve efficiency, and make better decisions.

### OBJECTIVE

The objective of the Hotel Management project is to provide a simple and user-friendly interface to manage hotel reservations and guest information. It aims to streamline the process of handling guest records, performing operations on the data, and generating visual representations of the data using graphs and charts.

The application will be able to perform a variety of operations on hotel reservations, including:

* Adding new records to the dataFrame
* Deleting records from the dataFrame
* Searching for records based on various columns, such as name, check-in date, check-out date, and room type
* Viewing the latest reservation
* Plotting a pie chart of guests by country or a bar graph of guests by check-in status
* Loading the dataFrame back to the CSV file

### LIMITATION

* The project is CLI-based. This means that it can only be used from the command line. Some users may prefer a graphical user interface (GUI).
* The project only works with CSV files. This means that the data must be in CSV format. Some hotels may use a different format for their reservation data.
* The project does not support all features of a commercial hotel management system. For example, the project does not support online booking or payments.

Despite these limitations, the hotel management project has the potential to be a valuable tool for hotels of all sizes. It can help to automate tasks, improve efficiency, and make better decisions.

### **SYSTEM CONFIGURATION**

The system configuration required to run the hotel management project is as follows:

**Hardware:**

A computer with a processor that is at least 1.5 GHz

At least 4 GB of RAM

At least 10 GB of free disk space

**Software:**

Python 3.11 or higher

pandas

Matplotlib

The project can be run on any operating system that supports Python, such as Windows, macOS, and Linux.

To install the required software, you can use the following commands:

pip install python

pip install pandas

pip install matplotlib

### **THEORETICAL BACKGROUND**

**1. What is Python Programming Language?**

Python is a high-level, interpreted, and general-purpose programming language. It was created by Guido van Rossum and first released in 1991. Python emphasizes code readability and simplicity, making it a great language for beginners. It has a large and active community, making it easy to find resources, libraries, and support.

Python is versatile and can be used for a wide range of applications, such as web development, data analysis, artificial intelligence, automation, and more.

**2. What is Pandas?**

Pandas is an open-source Python library built on top of NumPy. It provides data structures and functions for efficiently manipulating and analyzing large datasets. Pandas is widely used in data analysis and data science tasks.

**3. What is NumPy?**

NumPy (Numerical Python) is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays efficiently. NumPy is the foundation of many other scientific Python libraries, including Pandas.

**4. What is a Series and DataFrame?**

**- Series:** In Pandas, a Series is a one-dimensional labeled array that can hold data of any type (e.g., integers, strings, floats). It is similar to a Python list or array, but it has additional functionalities. A Series consists of two parts: the data and the index, which can be used to access the data.

- **DataFrame**: A DataFrame is a two-dimensional labeled data structure in Pandas, resembling a table or spreadsheet. It consists of rows and columns, where each column can have different data types. You can think of a DataFrame as a collection of Series. DataFrames are incredibly powerful for data manipulation, cleaning, and analysis.

**5. What is a CSV file?**

CSV stands for "Comma-Separated Values." It is a plain text file format used to store tabular data. Each line of the file represents a row, and the values within each row are separated by commas (or other delimiters like semicolons or tabs). CSV files are widely used for data storage and data exchange between different software systems.

**6. How to Load CSV into DataFrame using Pandas?**

Pandas provides a simple method to load CSV data into a DataFrame using the `pd.read\_csv()` function. Here's how you can do it:

```python

import pandas as pd

# Load CSV data into a DataFrame

df = pd.read\_csv('data.csv')

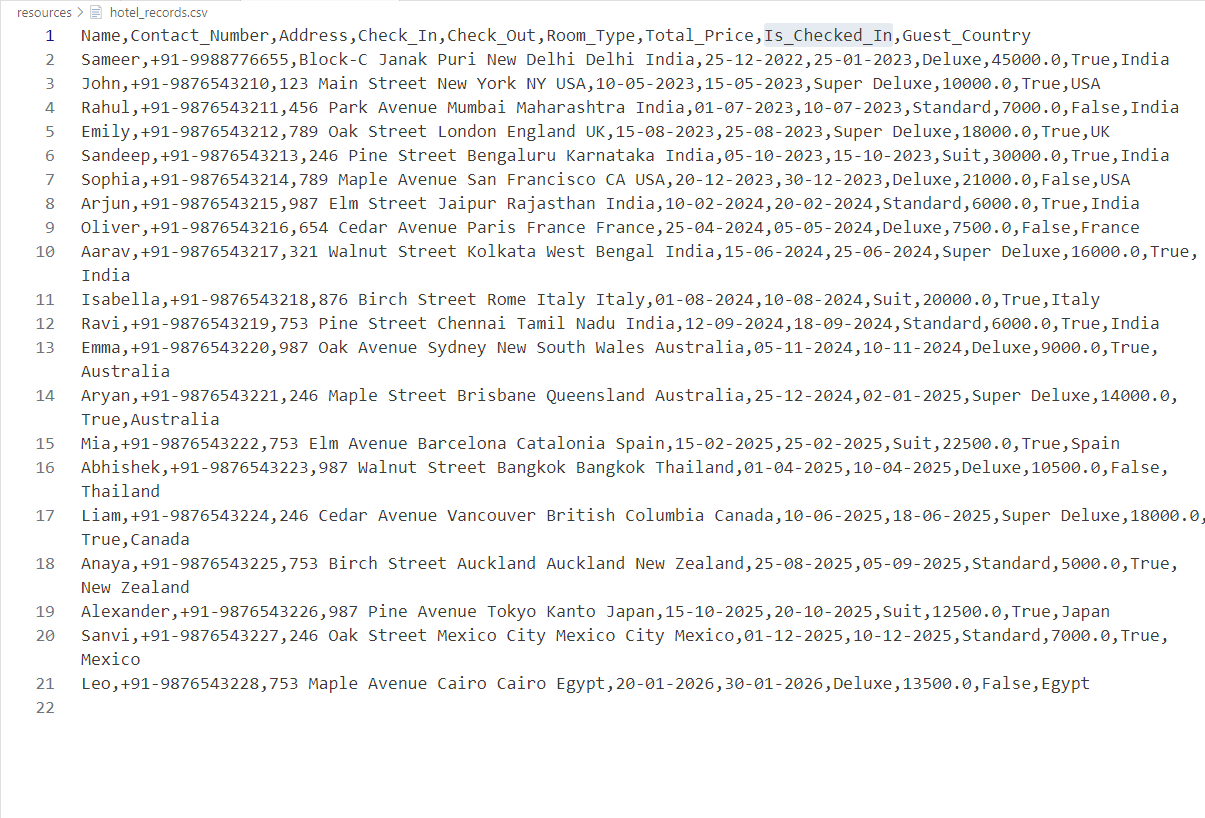
# Now, you can work with the DataFrame 'df'

# For example, you can print the first few rows

print(df.head())

```

### **HOTEL MANAGEMENT CSV USED**



### **SOURCE CODE**

### from init import loadDataFrameFromCsv

### from tasks import init\_execute\_task, taskList

### ACTIVE = True

### DF = None

### 

### 

### def main():

### global DF

### global ACTIVE

### print(" #### BOOTING UP THE APPLICATION ### ")

### print(" #### Hotel Management Software ### ")

### print(" #### loading csv data into the system ### ")

### DF = loadDataFrameFromCsv()

### print(" #### data has been successfully loaded into the system ### ")

### while ACTIVE:

### option = taskList()

### DF = init\_execute\_task(DF, option)

### print(" ############### ")

### if (type(DF) == type(False)):

### ACTIVE = False

### else:

### print("SHOW TASK MENU ?")

### choice = input("Y/N: ")

### if (choice.lower() == 'y'):

### ACTIVE = True

### else:

### ACTIVE = False

### 

### print(" #### SHUTING DOWN THE APPLICATION ### ")

### print(" #### Have A Nice Day :D ### ")

### 

### 

### if \_\_name\_\_ == '\_\_main\_\_':

### main()

import pandas as pd

CSV\_PATH = "./resources/hotel\_records.csv"

def loadDataFrameFromCsv():

try:

df = pd.read\_csv(CSV\_PATH)

except:

print("error loading the csv file path")

print(" program terminated ")

return df

def loadDataFrameToCsv(df):

try:

print(" #### Saving Records To The Csv #### ")

df.to\_csv(CSV\_PATH, index=False)

print(" #### records saved back to csv #### ")

except:

print("error loading the dataFrame to the csv file path")

print(" program terminated ")

return df

from insert\_record import insert\_record

from delete\_record import delete\_record

from get\_records import \*

from init import loadDataFrameToCsv

from plot import \*

def init\_execute\_task(df, option):

if (option == "1"):

return insert\_record(df)

elif (option == "2"):

column = 'Name'

return delete\_record(df, column)

elif (option == "3"):

return get\_latest\_reservation(df)

elif (option == "4"):

return get\_reservation\_by\_name(df)

elif (option == "5"):

return get\_reservation\_by\_check\_in\_date(df)

elif (option == "6"):

return get\_reservation\_by\_check\_out\_date(df)

elif (option == "7"):

return get\_all\_reservation\_checked\_in\_status(df, True)

elif (option == "8"):

return get\_all\_reservation\_checked\_in\_status(df, False)

elif (option == "9"):

return plot\_guests\_by\_country(df)

elif (option == "10"):

return plot\_guest\_by\_check\_in\_status(df)

elif (option == "11"):

return loadDataFrameToCsv(df)

elif (option == "12"):

print("Exiting...")

return False

else:

print("Invalid Option")

return False

def taskList():

TASK\_DICT = {

"1": "Making a new booking reservation",

"2": "Deleting a booking reservation",

"3": "See latest 5 reservations",

"4": "Search a booking using name",

"5": "Search a booking Using check-in date",

"6": "Search a booking Using check-out date",

"7": "See all guests who are checked in",

"8": "See all guests who are checked out",

"9": "Plot the pie chart of guest by the country",

"10": "Plot the bar graph of guest by the checked in status",

"11": "Save reservations back to csv",

"12": "Exit the application"

}

for key in TASK\_DICT:

print(f" {key} --- {TASK\_DICT[key]} ",)

operation\_selected = input("select the task : ")

if (not operation\_selected.isdigit() or (operation\_selected.lower() not in TASK\_DICT.keys())):

print(

f" #### INVALID TASK SELECTED {operation\_selected} ### ")

return "-1"

else:

return operation\_selected

from constants import CHECK\_IN\_COLUMN, CHECK\_OUT\_COLUMN

def insert\_record(df):

print(" ### Add new reservations record ### ")

input\_values = []

column\_list = list(df.columns)

for column in column\_list:

column\_name = str(column).replace("\_", " ")

if (column not in [CHECK\_OUT\_COLUMN, CHECK\_IN\_COLUMN]):

value = input(f"please provide {column\_name} :")

input\_values.append(value)

else:

value = input(

f"please provide {column\_name} in format dd-mm-yyyy :")

input\_values.append(value)

print(f" ### new reservation to be inserted in records is ### \n", input\_values)

try:

df.loc[df.index[-1]+1, :] = input\_values # statement to add new record

except:

print(" ### error while inserting record ### ")

exit()

print(" ### new reservation made successfully ### ")

print(" ### last two reservations are ### ")

print(df.tail(2))

return df

def delete\_record(df, column):

all\_names = list(df[column])

print(f" #### All Guests Name #### \n")

print(all\_names)

name = input(f"enter the {column} to delete its reservation: ")

if (name in all\_names):

try:

record\_index = df[df[column] == name].index

print(f"deleting the {str(record\_index)} record with name {name} ")

# axis 0 to drop a row/record

df.drop(record\_index, axis=0, inplace=True)

return df

except:

print(" error occurred while deleting record of index ",

str(record\_index))

exit()

else:

print(f" record with {name} not found")

return True

from constants import CHECK\_IN\_COLUMN, CHECK\_OUT\_COLUMN, NAME\_COLUMN, IS\_CHECKED\_IN\_COLUMN

def get\_latest\_reservation(df):

print(" ### the last 5 reservations are ### ")

print(df.tail(5))

return df

def get\_reservation\_by\_name(df):

print(" ### Search reservation record by name ### ")

all\_names = list(df[NAME\_COLUMN])

print(f" #### All Guests Name #### \n")

print(all\_names)

print("pls check the casing of the name")

name = input(f"enter the Name to check reservation: ")

if (name in all\_names):

try:

record = df[df[NAME\_COLUMN] == name]

print(f"Record Found")

print(record)

return df

except:

print(" error occurred while fetching record with name", name)

exit()

else:

print(f" record with {name} not found")

return df

def get\_reservation\_by\_check\_in\_date(df):

print(" ### Search reservations record by check in date ### ")

current\_check\_in\_dates = set(df[CHECK\_IN\_COLUMN])

print("All distinct check in dates in the system")

print(current\_check\_in\_dates)

query\_date = input(f"enter the check-in date in format dd-mm-yyyy: ")

if (query\_date in current\_check\_in\_dates):

print(

f" ### reservations by check in date {query\_date} found ### ")

records = df[df[CHECK\_IN\_COLUMN] == query\_date]

print(records)

else:

print(

f" ### reservation by check in date {query\_date} not found ### ")

return df

def get\_reservation\_by\_check\_out\_date(df):

print(" ### Search reservations record by check out date ### ")

query\_date = input(f"enter the check-out date in format dd-mm-yyyy: ")

current\_check\_out\_dates = list(df[CHECK\_OUT\_COLUMN])

if (query\_date in current\_check\_out\_dates):

print(

f" ### reservations by check out date {query\_date} found ### ")

records = df[df[CHECK\_OUT\_COLUMN] == query\_date]

print(records)

else:

print(

f" ### reservation by check out date {query\_date} not found ### ")

return df

def get\_all\_reservation\_checked\_in\_status(df, checked\_in\_status):

if checked\_in\_status:

print(" ### all checked in reservations ### ")

else:

print(" ### all checked in reservations ### ")

records = df[df[IS\_CHECKED\_IN\_COLUMN] == checked\_in\_status]

print(records)

return df

import pandas as pd

import matplotlib.pyplot as plt

from constants import GUEST\_COUNTRY, IS\_CHECKED\_IN

def plot\_guests\_by\_country(data\_frame):

print(" #### Plotting the pie chart of guest by the country ### ")

try:

country\_counts = data\_frame[GUEST\_COUNTRY].value\_counts()

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

plt.pie(country\_counts, labels=country\_counts.index, autopct='%1.1f%%')

plt.title('Guests Residing in the Hotel by Country')

plt.show()

except:

print(" #### ERROR ! Plotting the pie chart of guest by the country ### ")

return data\_frame

def plot\_guest\_by\_check\_in\_status(df):

print(" #### Plotting the bar graph of guest by the checked in status ### ")

try:

total\_guest = len(df)

guests\_checked\_in = len(df[df["Is\_Checked\_In"] == True])

guests\_not\_checked\_in = total\_guest-guests\_checked\_in

x\_ticks\_labels = ["Total Guests","Checked In Guests", "Checked Out Guests"]

data = [total\_guest, guests\_checked\_in, guests\_not\_checked\_in]

colors = ['orange', 'green', 'purple']

plt.bar(x\_ticks\_labels,data, data=data, color=colors)

plt.xlabel("Guests Status")

plt.ylabel("Guests Count By Status")

plt.title("Guests Status")

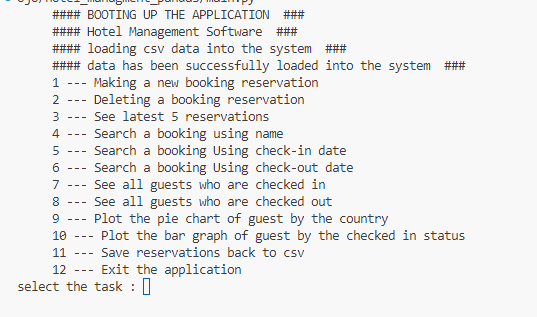
plt.show()

except:

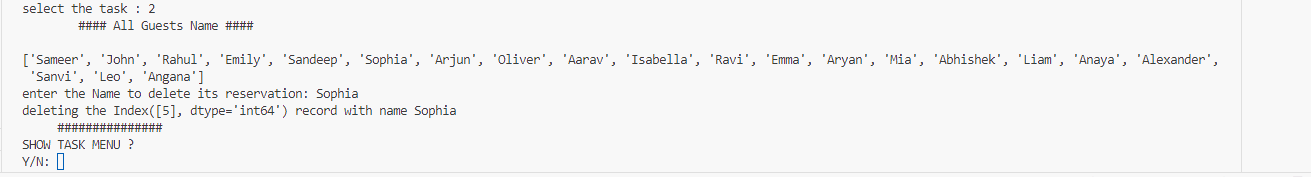
print(" #### ERROR ! Plotting the bar graph of guest by the checked in status ### ")

return df

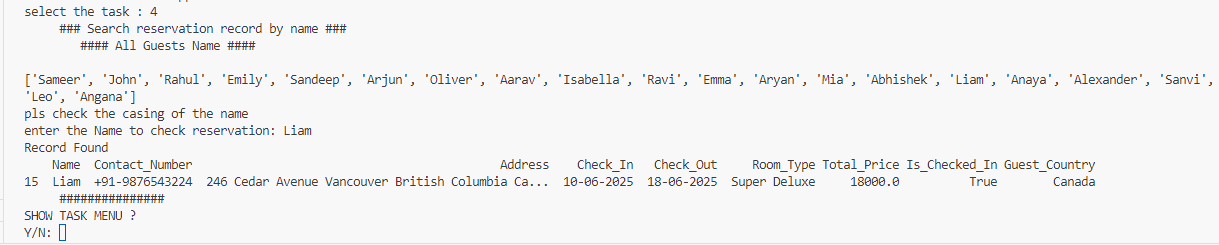
### **OUTPUTS**

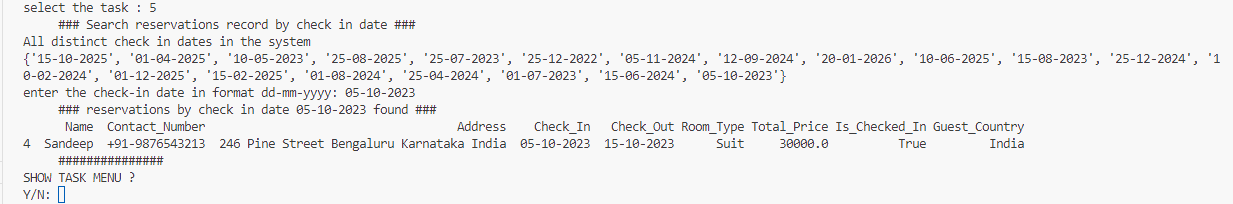
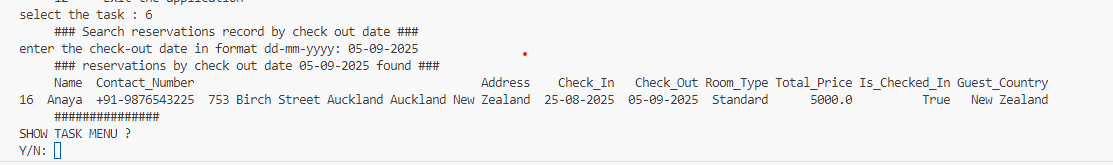


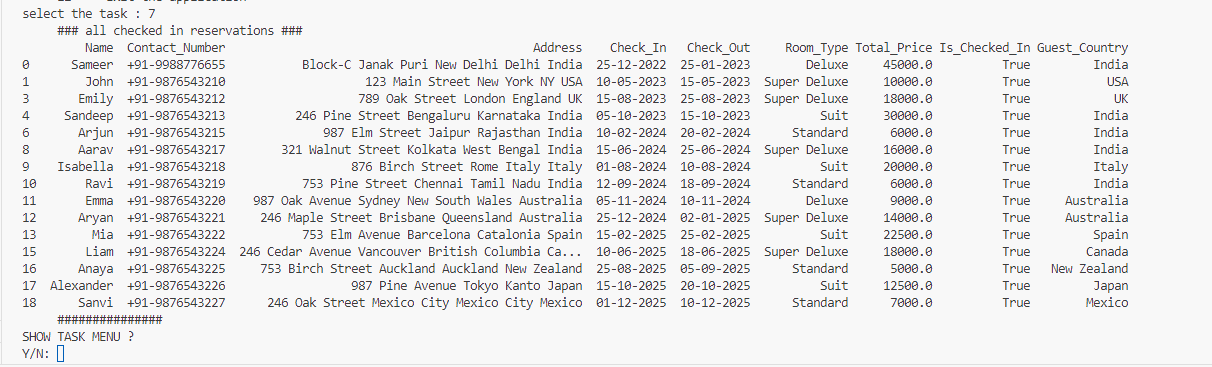
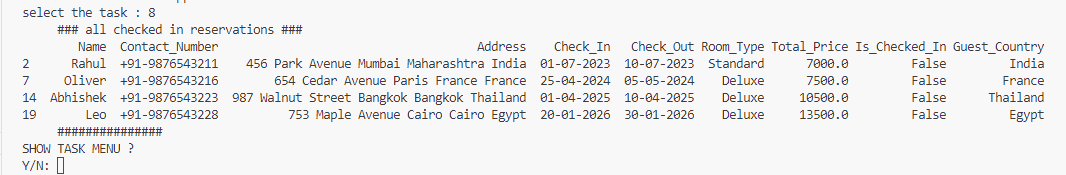


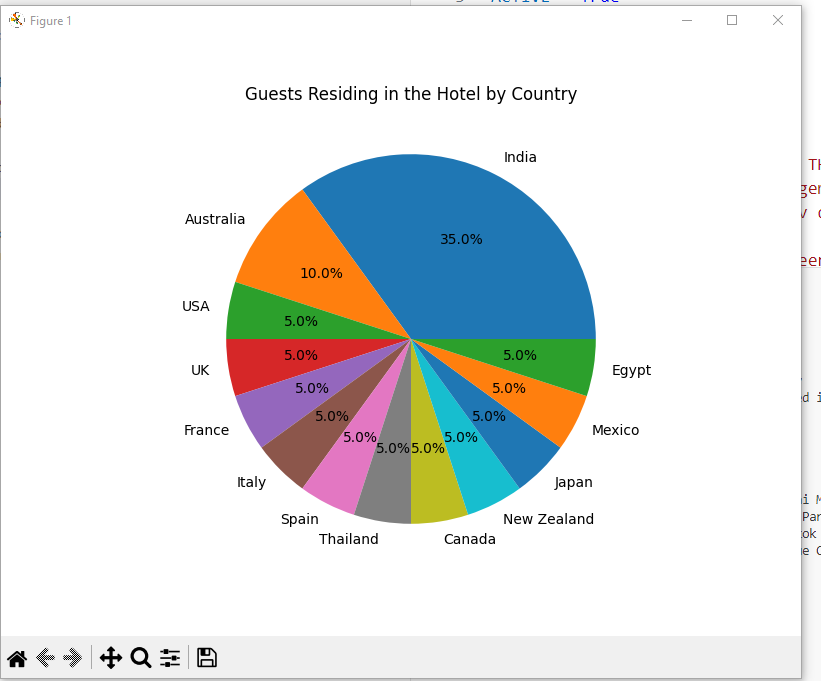


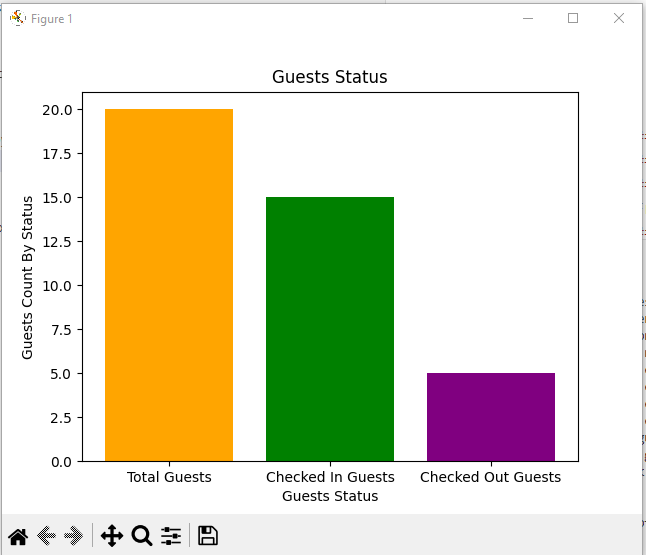
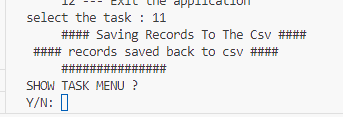







### 

### 