In [1]:
```
# Import all required librery

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Read csv file
Data= pd.read_csv("customer_booking.csv",encoding="latin1")
```

In [3]:
```
# check basic information
```

In [4]:
```
Data.head(10)
```

Out[4]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	fli
0	2	Internet	RoundTrip	262	19	7	
1	1	Internet	RoundTrip	112	20	3	
2	2	Internet	RoundTrip	243	22	17	
3	1	Internet	RoundTrip	96	31	4	
4	2	Internet	RoundTrip	68	22	15	
5	1	Internet	RoundTrip	3	48	20	
6	3	Internet	RoundTrip	201	33	6	
7	2	Internet	RoundTrip	238	19	14	
8	1	Internet	RoundTrip	80	22	4	
9	1	Mobile	RoundTrip	378	30	12	

In [5]:
```
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   num_passengers         50000 non-null  int64  
 1   sales_channel          50000 non-null  object 
 2   trip_type              50000 non-null  object 
 3   purchase_lead          50000 non-null  int64  
 4   length_of_stay         50000 non-null  int64  
 5   flight_hour            50000 non-null  int64  
 6   flight_day             50000 non-null  object 
 7   route                  50000 non-null  object 
 8   booking_origin         50000 non-null  object 
 9   wants_extra_baggage    50000 non-null  int64  
 10  wants_preferred_seat   50000 non-null  int64  
 11  wants_in_flight_meals  50000 non-null  int64  
 12  flight_duration        50000 non-null  float64
 13  booking_complete       50000 non-null  int64  
dtypes: float64(1), int64(8), object(5)
memory usage: 5.3+ MB
```

In [6]:
```
Data.isnull().sum()
```

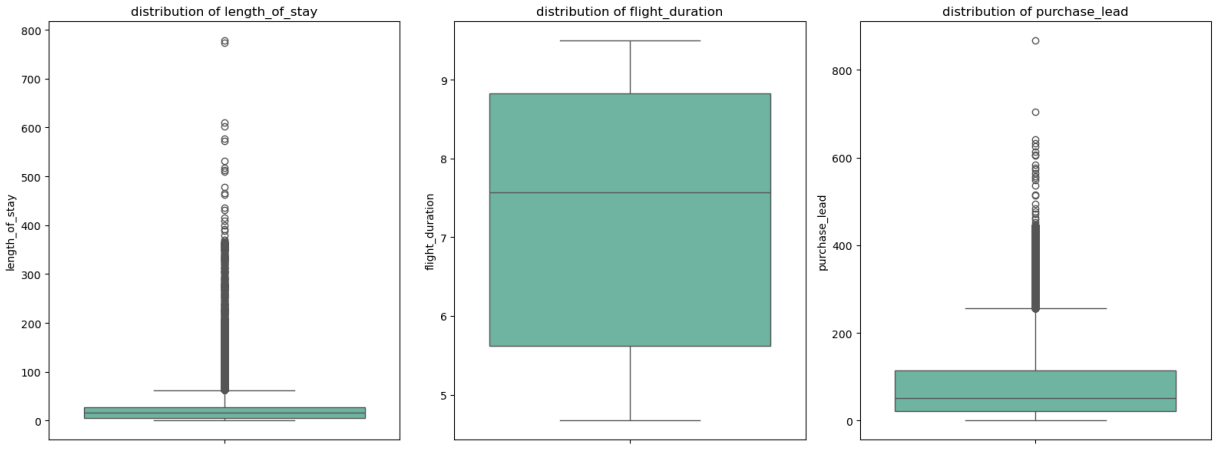
Out[6]:
```
num_passengers          0
sales_channel           0
trip_type               0
purchase_lead           0
length_of_stay          0
flight_hour             0
flight_day              0
route                   0
booking_origin          0
wants_extra_baggage     0
wants_preferred_seat    0
wants_in_flight_meals   0
flight_duration         0
booking_complete        0
dtype: int64
```

In [7]:
```
Data.describe()
```

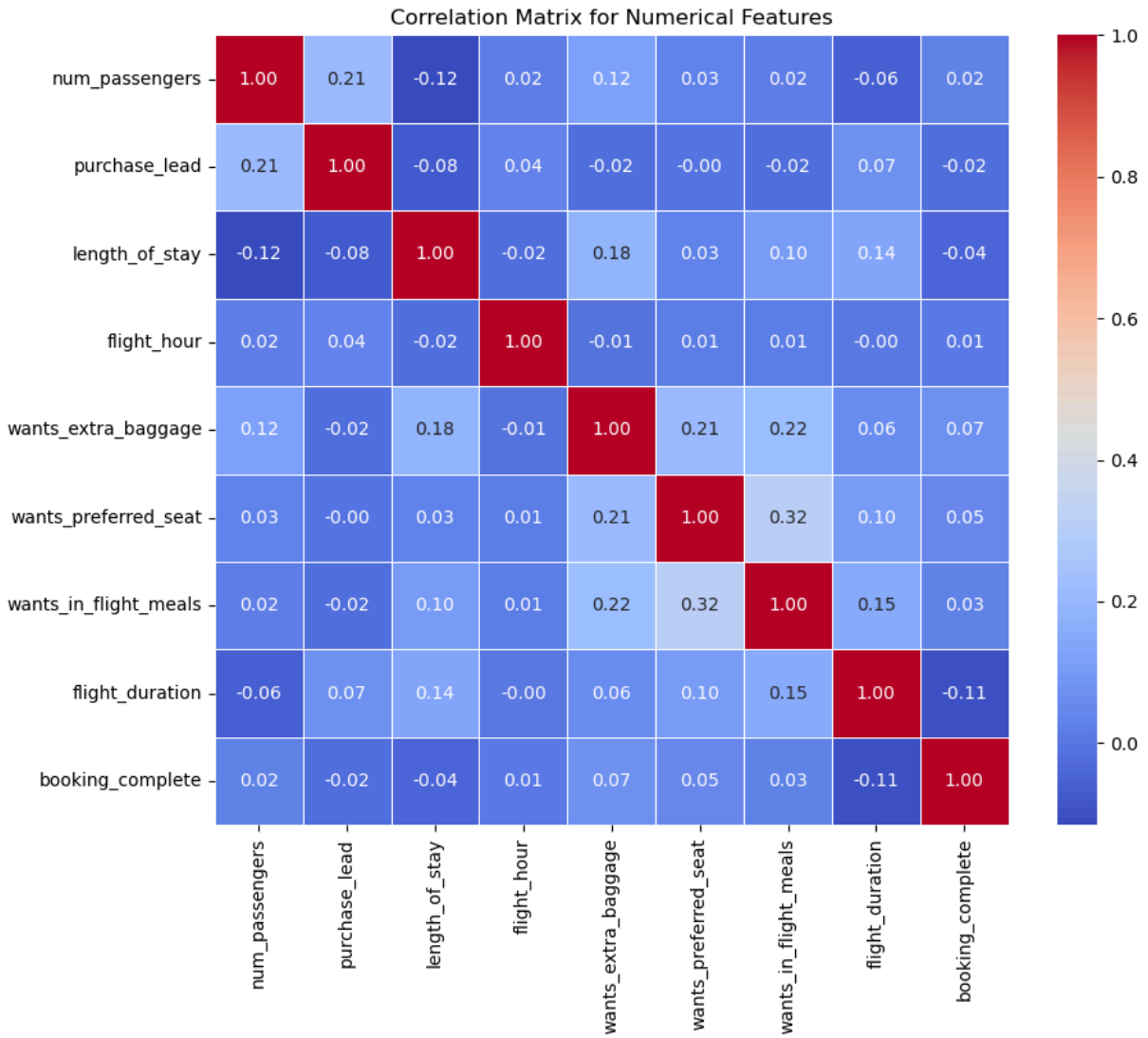
Out[7]:

	num_passengers	purchase_lead	length_of_stay	flight_hour	wants_extra_baggage
count	50000.000000	50000.000000	50000.00000	50000.00000	50000.000000
mean	1.591240	84.940480	23.04456	9.06634	0.668780
std	1.020165	90.451378	33.88767	5.41266	0.470657
min	1.000000	0.000000	0.00000	0.00000	0.000000
25%	1.000000	21.000000	5.00000	5.00000	0.000000
50%	1.000000	51.000000	17.00000	9.00000	1.000000
75%	2.000000	115.000000	28.00000	13.00000	1.000000
max	9.000000	867.000000	778.00000	23.00000	1.000000

In [8]:
```
# Data Visualisation

Numerical_column= ['length_of_stay', 'flight_duration', 'purchase_lead']

plt.figure(figsize=(16,6))
for i, column in enumerate(Numerical_column,1):
    plt.subplot(1,3,i)
    sns.boxplot(data=Data,y=column,palette='Set2')
    plt.title(f'distribution of {column}')
    plt.tight_layout()
plt.show()
```



In [9]:
```
# Calculate correlation matrix
correlation_matrix = Data.corr(numeric_only=True)

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=
plt.title('Correlation Matrix for Numerical Features')
plt.show()

# Sort correlation values with respect to the target variable
correlation_with_target = correlation_matrix['booking_complete'].sort_values(ascend
correlation_with_target
```



Out[9]:
```
booking_complete          1.000000
wants_extra_baggage       0.068139
wants_preferred_seat      0.050116
wants_in_flight_meals     0.026511
num_passengers            0.024116
flight_hour               0.007127
purchase_lead            -0.022131
length_of_stay           -0.042408
flight_duration          -0.106266
Name: booking_complete, dtype: float64
```

In [10]:
```
# Feature Engineering and Preprocessing
for column in ['length_of_stay', 'flight_duration', 'purchase_lead']:
    lower_limit = Data[column].quantile(0.01)  # 1st percentile
    upper_limit = Data[column].quantile(0.99)  # 99th percentile
    Data[column] = Data[column].clip(lower=lower_limit, upper=upper_limit)
```

In [11]:
```
# One-Hot Encoding for Categorical Variables
categorical_columns = ['sales_channel', 'trip_type', 'flight_day', 'booking_origin'
encoder = OneHotEncoder(sparse_output=False, drop='first')
encoded_cats = encoder.fit_transform(Data[categorical_columns])
encoded_cats_df = pd.DataFrame(encoded_cats, columns=encoder.get_feature_names_out(
```

In [12]:
```
# Scaling Numerical Features
numerical_columns = ['length_of_stay', 'flight_duration', 'purchase_lead', 'num_pas
scaler = StandardScaler()
scaled_nums = scaler.fit_transform(Data[numerical_columns])
scaled_nums_df = pd.DataFrame(scaled_nums, columns=numerical_columns)
```

In [13]:
```
# Combine encoded categorical and scaled numerical features
X = pd.concat([encoded_cats_df, scaled_nums_df], axis=1)

# Target variable
y = Data['booking_complete']
```

In [14]:
```
# Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Output the shapes of the train/test datasets
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[14]: ((40000, 116), (10000, 116), (40000,), (10000,))

In [15]:
```
# Initialize the Logistic Regression model
logistic_model = LogisticRegression(random_state=42, max_iter=1000)

# Train the model on the training data
logistic_model.fit(X_train, y_train)

# Predict on the test data
y_pred = logistic_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(accuracy)
```

```
0.8504
```

In [16]:
```
print(classification_rep)
```

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92      8504
           1       0.00      0.00      0.00      1496

    accuracy                           0.85     10000
   macro avg       0.43      0.50      0.46     10000
weighted avg       0.72      0.85      0.78     10000
```

In [17]:
```
print(conf_matrix)
```

```
[[8504    0]
 [1496    0]]
```