num_passengers = The number of passengers associated with each booking.

sales_channel = How customer reached to our website.

trip_type = Specifies whether the booking is for a one-way or round trip(Round Trip, One Way)

purchase_lead = number of days between travel date and booking date

length_of_stay = number of days spent at destination

flight_hour = hour of flight departure

flight_day = day of week of flight departure

route = origin -> destination flight route

booking_origin = country from where booking was made

wants_extra_baggage = if the customer wanted extra baggage in the booking

wants_preferred_seat = if the customer wanted a preferred seat in the booking

wants_in_flight_meals = if the customer wanted in-flight meals in the booking

flight_duration = total duration of flight (in hours)

booking_complete = flag indicating if the customer completed the booking

Importing necessary libraries

In [1]:

```
import numpy as np
import pandas as pd                                    # data processing
import matplotlib.pyplot as plt                        # visualization
import seaborn as sns
```

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
```

Load the dataset

In [3]:

```
passenger_df = pd.read_csv('D:/MDA 4th Sem/ML/Passanger_booking_data.csv')
passenger_df.head(5)
```

Out[3]:

assengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	flight_day	
1	Internet	RoundTrip	21	12	6	Tue	AKL
2	Internet	RoundTrip	262	19	7	Sat	AK
1	Internet	RoundTrip	112	20	3	Sat	AK
2	Internet	RoundTrip	243	22	17	Wed	AK
1	Internet	RoundTrip	96	31	4	Sat	AK

In [4]:

```
passenger_df.shape
```

Out[4]:

```
(50002, 14)
```

There are 50002 rows and 14 columns in the dataframe.

In [5]:

```
passenger_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50002 entries, 0 to 50001
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   num_passengers         50002 non-null  int64  
 1   sales_channel          50002 non-null  object 
 2   trip_type              50002 non-null  object 
 3   purchase_lead          50002 non-null  int64  
 4   length_of_stay         50002 non-null  int64  
 5   flight_hour            50002 non-null  int64  
 6   flight_day             50002 non-null  object 
 7   route                  50002 non-null  object 
 8   booking_origin         50002 non-null  object 
 9   wants_extra_baggage    50002 non-null  int64  
 10  wants_preferred_seat   50002 non-null  int64  
 11  wants_in_flight_meals  50002 non-null  int64  
 12  flight_duration        50002 non-null  float64
 13  booking_complete       50002 non-null  int64  
dtypes: float64(1), int64(8), object(5)
memory usage: 5.3+ MB
```

It seems that there are no missing values, which will be checked in more detail below. We have 14 features in which the target variable is booking complete. Among the variables there are 5 categorical variables and 9 numerical variables.

In [6]:

```
# selecting the categorical variables
categorical_var = passenger_df.select_dtypes(include='object')
# Obtaining summary statistics for the categorical variables
categorical_stat = categorical_var.describe().T
categorical_stat
```

Out[6]:

	count	unique	top	freq
sales_channel	50002	2	Internet	44383
trip_type	50002	3	RoundTrip	49498
flight_day	50002	7	Mon	8102
route	50002	799	AKLKUL	2680
booking_origin	50002	104	Australia	17873

In [7]:

```
# selecting numerical variables
numerical_var = passenger_df.select_dtypes(exclude='object')
# Obtaining summar statistics for the numerical variables
numerical_stat = numerical_var.describe().T
numerical_stat
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
num_passengers	50002.0	1.591256	1.020167	1.00	1.00	1.00	2.00	9.0
purchase_lead	50002.0	84.940582	90.450548	0.00	21.00	51.00	115.00	867.0
length_of_stay	50002.0	23.044778	33.887171	0.00	5.00	17.00	28.00	778.0
flight_hour	50002.0	9.066277	5.412569	0.00	5.00	9.00	13.00	23.0
wants_extra_baggage	50002.0	0.668773	0.470659	0.00	0.00	1.00	1.00	1.0
wants_preferred_seat	50002.0	0.296968	0.456927	0.00	0.00	0.00	1.00	1.0
wants_in_flight_meals	50002.0	0.427143	0.494668	0.00	0.00	0.00	1.00	1.0
flight_duration	50002.0	7.277524	1.496854	4.67	5.62	7.57	8.83	9.5
booking_complete	50002.0	0.149574	0.356657	0.00	0.00	0.00	0.00	1.0

Checking for null values

In [8]:

```
# Check missing value
passenger_df.isnull().sum()
```

Out[8]:

```
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
```

There is no missing values in the dataframe.

Checking for duplicate values

In [9]:

```
passenger_df.duplicated().sum()
```

Out[9]:

```
719
```

We have 719 duplicate values, so we need to remove those values.

In [10]:

```
# Remove duplicate rows
passenger_df = passenger_df.drop_duplicates()
passenger_df.duplicated().sum()
```

Out[10]:

```
0
```

In [11]:

```
passenger_df.shape
```

Out[11]:

```
(49283, 14)
```

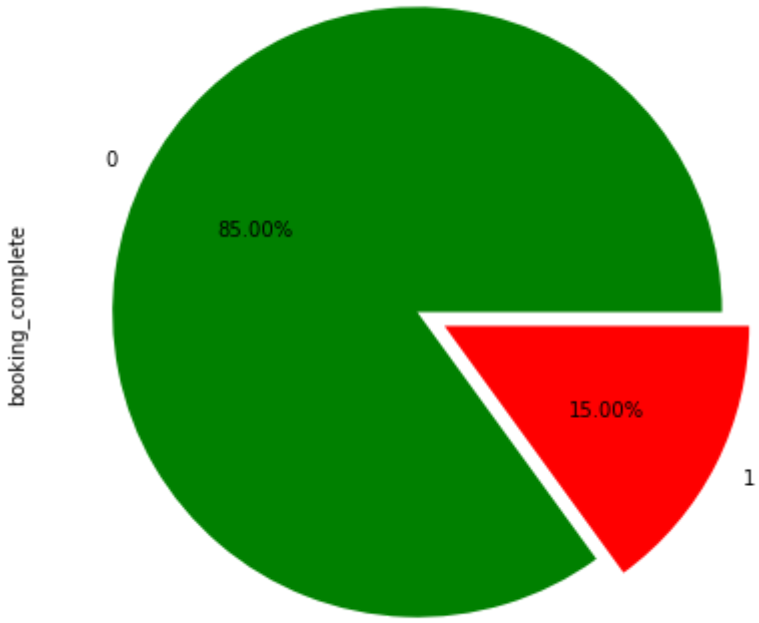
Now we have 49283 rows.

In [12]:

```
plt.figure(figsize = (7,7))
passenger_df['booking_complete'].value_counts().plot(kind = 'pie', autopct = '%.2f%%', co
```

Out[12]:

```
<AxesSubplot:ylabel='booking_complete'>
```



From the above plot it is clear that 85% of the passengers have not booked. Only 15% of the passengers completed the booking.

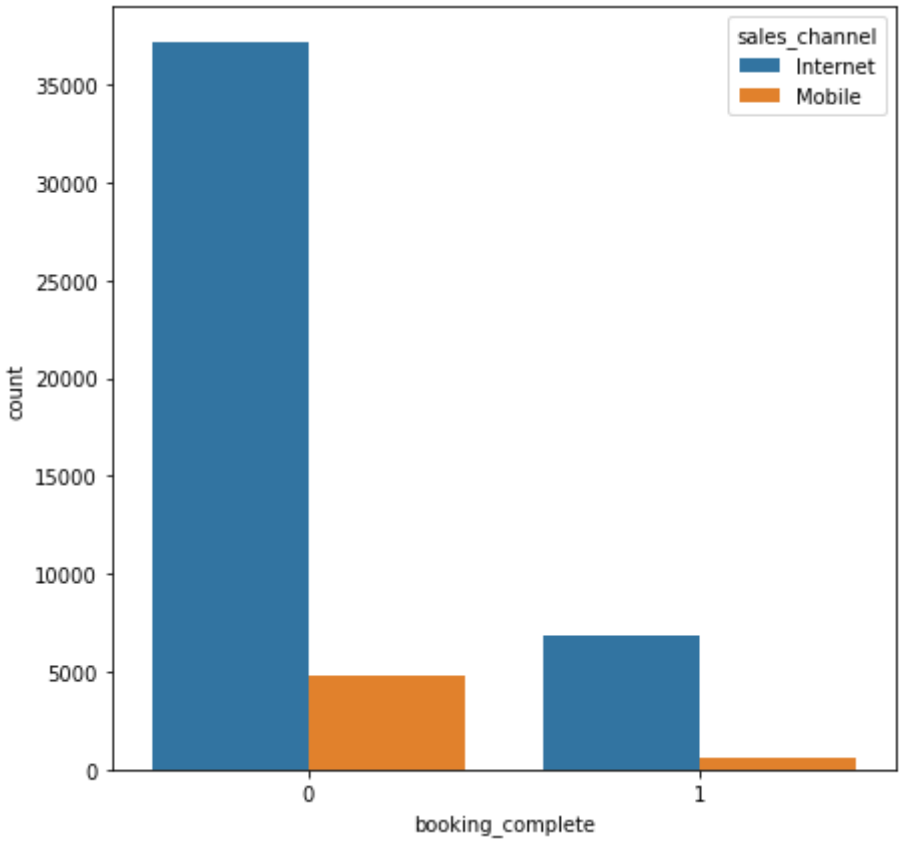
Internet booking vs Mobile booking

In [31]:

```
plt.figure(figsize = (7,7))
sns.countplot(x='booking_complete', data=passenger_df, hue='sales_channel')
```

Out[31]:

```
<AxesSubplot:xlabel='booking_complete', ylabel='count'>
```



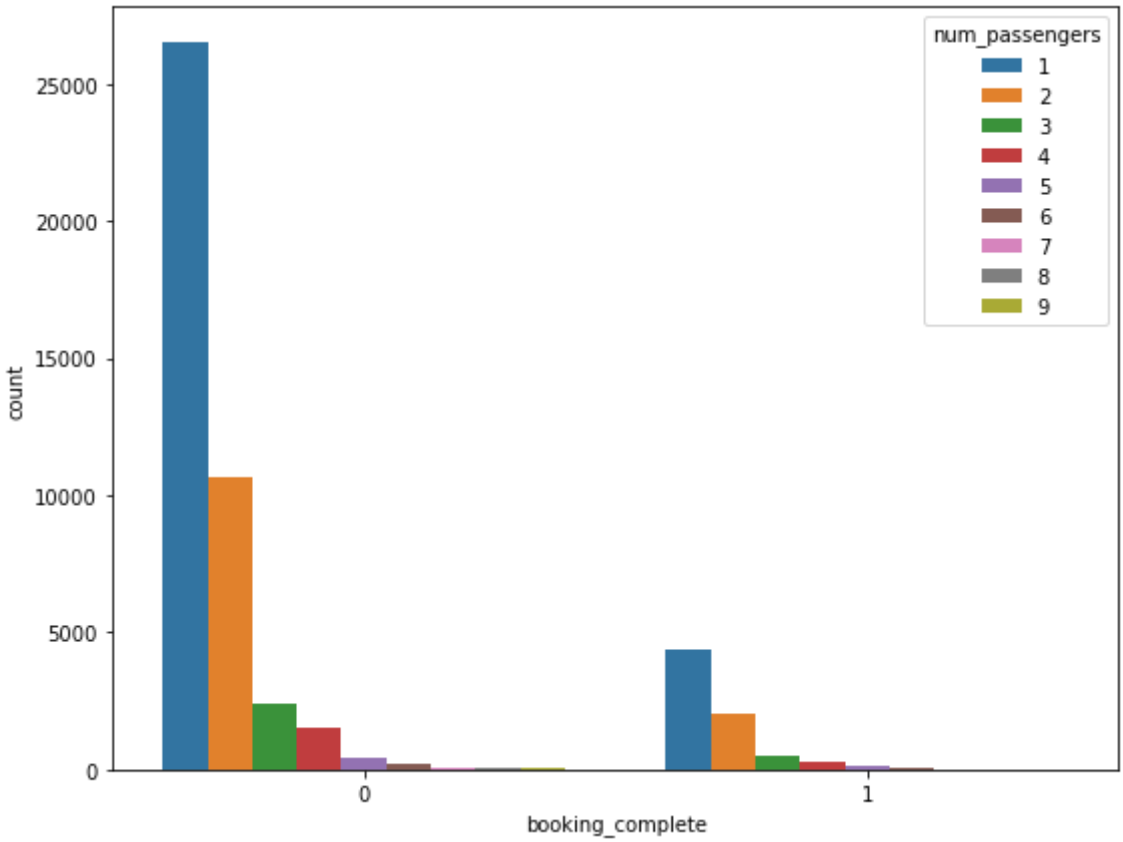
There is more internet booking than the mobile booking.

In [30]:

```
plt.figure(figsize = (9,7))
sns.countplot(x='booking_complete', data=passenger_df, hue='num_passengers')
```

Out[30]:

```
<AxesSubplot:xlabel='booking_complete', ylabel='count'>
```



Mostly 1 passenger has booked the flight. As the number of passengers increases the booking made by the passengers decreases.

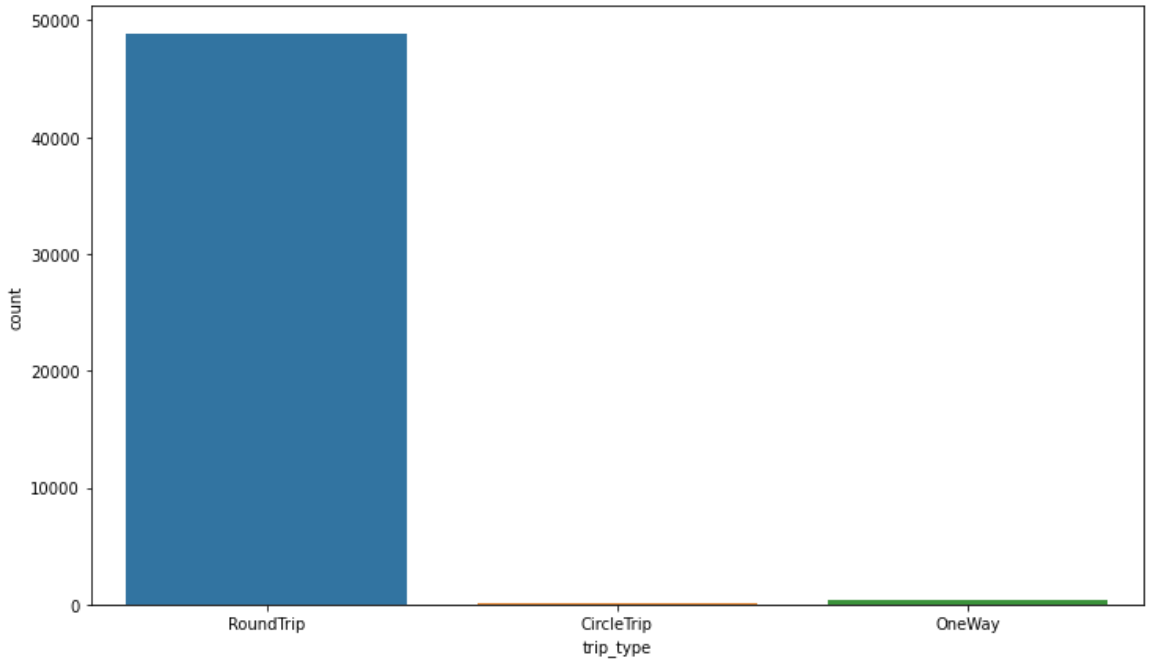
Prefered type of trip

In [39]:

```
plt.figure(figsize = (12,7))
sns.countplot(x='trip_type', data=passenger_df)
```

Out[39]:

```
<AxesSubplot:xlabel='trip_type', ylabel='count'>
```



Most of the people prefered Roundtrip as the type of trip.

In [40]:

```
passenger_df.groupby('booking_complete')['flight_day'].value_counts()
```

Out[40]:

```
booking_complete  flight_day
0                 Mon           6794
                  Tue           6442
                  Wed           6333
                  Thu           6214
                  Fri           5709
                  Sun           5531
                  Sat           4868
1                 Wed           1230
                  Mon           1194
                  Tue           1117
                  Thu           1109
                  Fri            976
                  Sun            911
                  Sat            855
Name: flight_day, dtype: int64
```

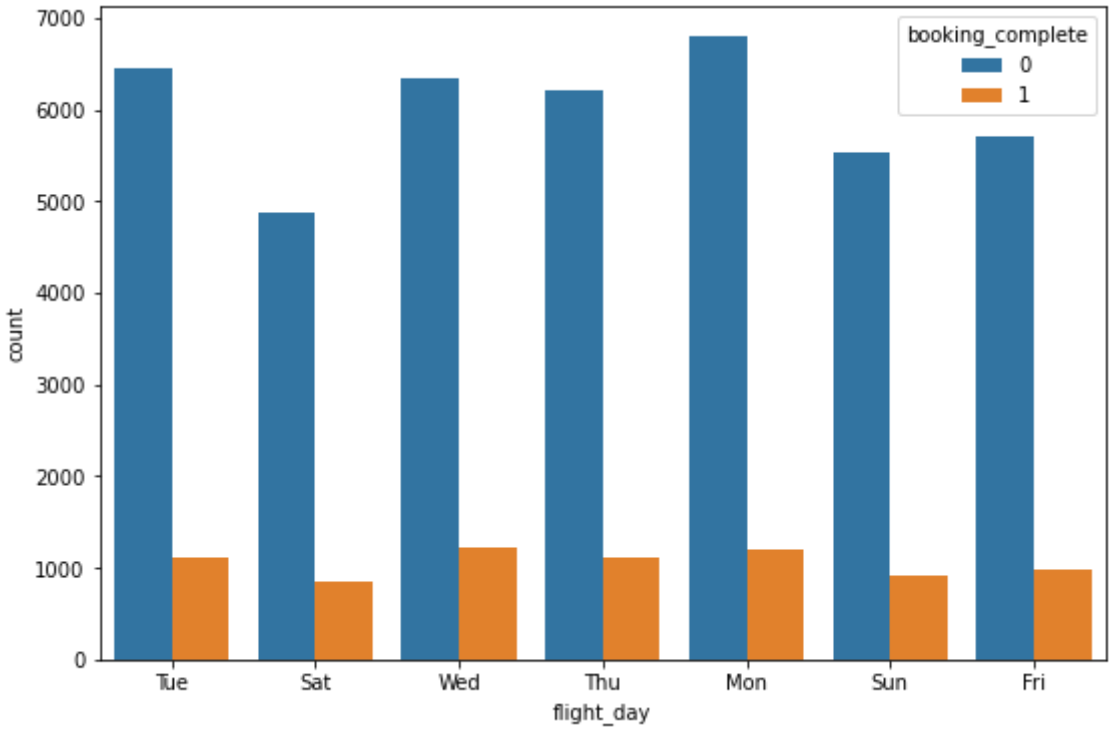
Most of the bookings happened on wednesday, and a few were made on saturday.

In [45]:

```
plt.figure(figsize = (9,6))
sns.countplot(data = passenger_df, x ='flight_day', hue = 'booking_complete')
```

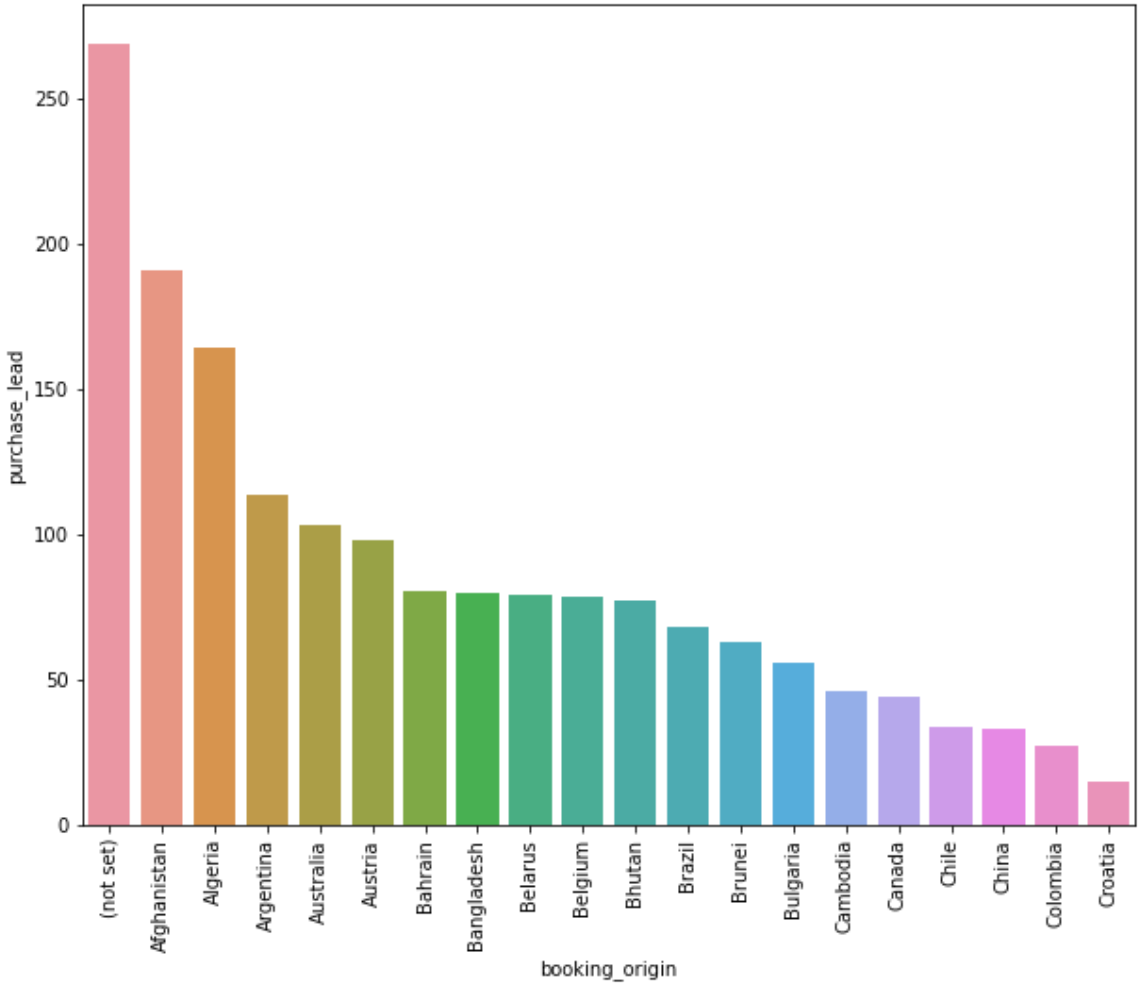
Out[45]:

```
<AxesSubplot:xlabel='flight_day', ylabel='count'>
```



In [55]:

```
plt.figure(figsize = (10,8))
df=passenger_df.groupby('booking_origin').agg({'purchase_lead':'mean'}).head(20)
df_sorted = df.sort_values(by='purchase_lead', ascending=False)
sns.barplot(x=df.index,y='purchase_lead',data=df_sorted)
plt.xticks(rotation=90)
plt.show()
```



In [59]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
passenger_df['sales_channel'] = le.fit_transform(passenger_df['sales_channel']).astype('i
passenger_df['trip_type'] = le.fit_transform(passenger_df['trip_type']).astype('int64')
passenger_df['flight_day'] = le.fit_transform(passenger_df['flight_day']).astype('int64')
passenger_df['route'] = le.fit_transform(passenger_df['route']).astype('int64')
passenger_df['booking_origin'] = le.fit_transform(passenger_df['booking_origin']).astype(
```

In [60]:

```
passenger_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49283 entries, 0 to 50001
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   num_passengers         49283 non-null  int64  
 1   sales_channel          49283 non-null  int64  
 2   trip_type              49283 non-null  int64  
 3   purchase_lead          49283 non-null  int64  
 4   length_of_stay         49283 non-null  int64  
 5   flight_hour            49283 non-null  int64  
 6   flight_day             49283 non-null  int64  
 7   route                  49283 non-null  int64  
 8   booking_origin         49283 non-null  int64  
 9   wants_extra_baggage    49283 non-null  int64  
 10  wants_preferred_seat   49283 non-null  int64  
 11  wants_in_flight_meals  49283 non-null  int64  
 12  flight_duration        49283 non-null  float64
 13  booking_complete       49283 non-null  int64  
dtypes: float64(1), int64(13)
memory usage: 5.6 MB
```

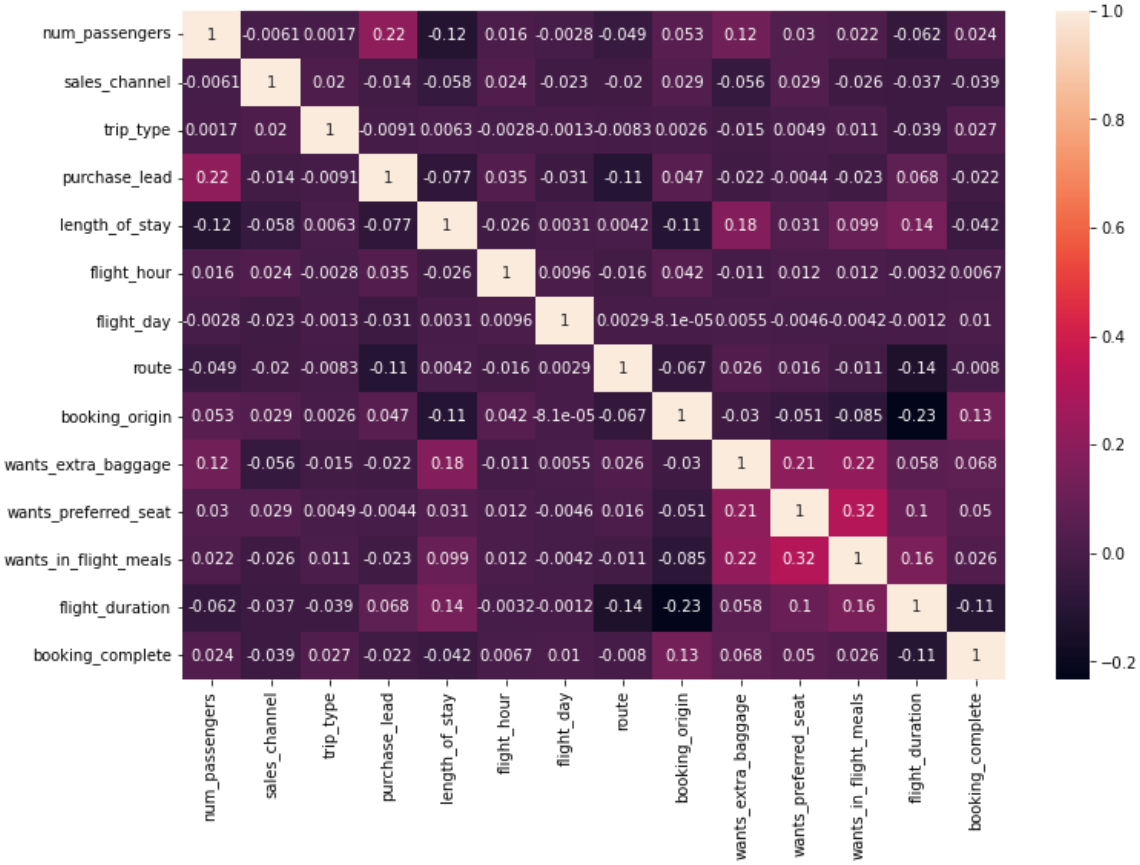
Correlation Matrix

In [86]:

```
plt.figure(figsize=(12, 8))
correlation_matrix = passenger_df.corr()
sns.heatmap(correlation_matrix, annot=True)
```

Out[86]:

<AxesSubplot:>



In [61]:

```
X = passenger_df.drop('booking_complete', axis=1)
Y = passenger_df['booking_complete']
```

In [89]:

```
from sklearn.feature_selection import mutual_info_classif
mi_score = mutual_info_classif(X,Y)
mi_score = pd.Series(mi_score, name="Mutual information Scores", index = X.columns)
mi_score = mi_score.sort_values(ascending=False)
mi_score
```

Out[89]:

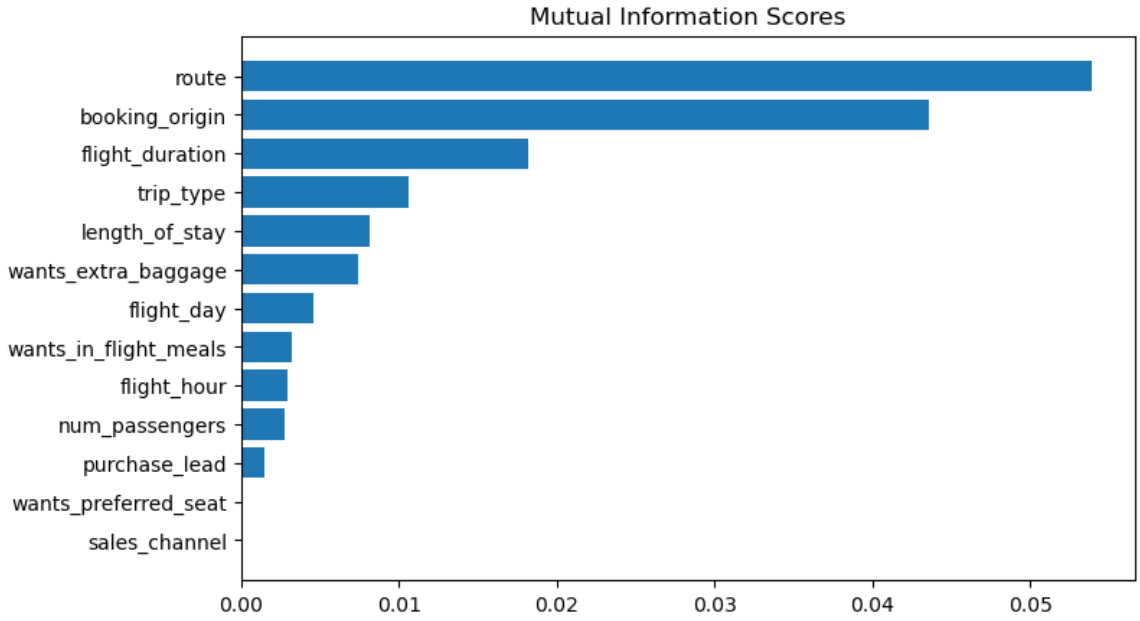
```
route                   0.053899
booking_origin          0.043528
flight_duration         0.018153
trip_type               0.010606
length_of_stay          0.008182
wants_extra_baggage     0.007415
flight_day              0.004623
wants_in_flight_meals   0.003195
flight_hour             0.002988
num_passengers          0.002800
purchase_lead           0.001504
wants_preferred_seat    0.000095
sales_channel           0.000000
Name: Mutual information Scores, dtype: float64
```

Mutual information score is a measure of the strength of association between feature and the target. The MI score will fall in the range from 0 to ∞. The higher value, the closer connection between this feature and the target.

In [92]:

```
def plot_mi_scores(scores):
    scores = scores.sort_values(ascending=True)
    width = np.arange(len(scores))
    ticks = list(scores.index)
    plt.barh(width, scores)
    plt.yticks(width, ticks)
    plt.title("Mutual Information Scores")

plt.figure(dpi=100, figsize=(8, 5))
plot_mi_scores(mi_score)
```



We can see route, booking_origin, flight_duration, length_of_stay, wants_extra_baggage are the top 5 features which are dependant with booking_complete feature

In [62]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state = 4
```

Logistic Regression

In [81]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
# Create a logistic regression model
model = LogisticRegression()
# train the model
model.fit(X_train, Y_train)
# Use the trained model to make predictions on the test data
Y_pred = model.predict(X_test)
#evaluate the model
accuracy = accuracy_score(Y_test, Y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8528238079134258
```

The overall accuracy of the model is 0.85. It represents the proportion of correctly classified instances (both true positives and true negatives) among all instances.

In [82]:

```
from sklearn.metrics import confusion_matrix
Confusion_matrix = confusion_matrix(Y_test, Y_pred)
print(Confusion_matrix)
```

```
[[12609     0]
 [ 2176     0]]
```

In [84]:

```
from sklearn.metrics import classification_report
print(classification_report(Y_test,Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92     12609
           1       0.00      0.00      0.00      2176

    accuracy                           0.85     14785
   macro avg       0.43      0.50      0.46     14785
weighted avg       0.73      0.85      0.79     14785
```

Precision measures the proportion of true positive predictions (correctly predicted bookings completed) among all positive predictions (total predicted bookings completed). For class 0, the precision is 0.85, meaning that out of all the predicted bookings not completed, 85% were correct. However, for class 1, the precision is 0.00, indicating that there were no correct predictions for bookings completed.

Recall measures the proportion of true positive predictions (correctly predicted bookings completed) among all actual positive instances (total actual bookings completed). For class 0, the recall is 1.00, indicating that all actual bookings not completed were correctly predicted. However, for class 1, the recall is 0.00, meaning that none of the actual bookings completed were correctly predicted.

The F1-score is the harmonic mean of precision and recall. It balances both metrics and is useful when dealing with imbalanced datasets. The F1-score for class 0 is 0.92, and for class 1, it is 0.00.