PREDICTING CUSTOMER BUYING BEHAVIOUR

The task is to build a predictive model and then to understand which factors influence customer booking.

In [69]:
```
#import libraries

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [70]:
```
#import the csv file

data=pd.read_csv("E:/customer.csv", encoding='ISO-8859-1')
data.head()
```

Out[70]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	flight_(
0	2	Internet	RoundTrip	262	19	7	
1	1	Internet	RoundTrip	112	20	3	
2	2	Internet	RoundTrip	243	22	17	V
3	1	Internet	RoundTrip	96	31	4	
4	2	Internet	RoundTrip	68	22	15	V

Explanatory Variables

- `num_passengers` = number of passengers travelling.
- `sales_channel` = sales channel booking was made on(internet, phone call).
- `trip_type` = trip Type (Round Trip, One Way, Circle Trip).
- `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, 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.

Predictive Variable

- `booking_complete` = flag indicating if the customer completed the booking(Binary:"1" means completed, "0" means not completed).

Exploratory Data Analysis

In [71]:
```
# view columns name
data.columns
```

Out[71]:
```
Index(['num_passengers', 'sales_channel', 'trip_type', 'purchase_lead',
       'length_of_stay', 'flight_hour', 'flight_day', 'route',
       'booking_origin', 'wants_extra_baggage', 'wants_preferred_seat',
       'wants_in_flight_meals', 'flight_duration', 'booking_complete'],
      dtype='object')
```

In [72]:
```
# information of the data
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
```

Check Column's Values

In [73]:
```
#view the categorical variables
#Nominal Data
print(data['sales_channel'].unique())
print(data['trip_type'].unique())
```

```
['Internet' 'Mobile']
['RoundTrip' 'CircleTrip' 'OneWay']
```

- `Internet` --> Booking through a Internet.
- `Mobile` --> Booking with a help of Phone Call.
- `Roundtrip` -->A ticket that allows a person to travel to one place and then return back to the place he or she left.
- `Onewaytrip` -->the journey back from a destination.
- `Circletrip` --> A circle trip is a return trip that usually includes multiple stops along the route of travel before returning to the point of origin.

In [74]:
```
# the day in which flight departure
print(data.flight_day.unique())
```

```
['Sat' 'Wed' 'Thu' 'Mon' 'Sun' 'Tue' 'Fri']
```

In [75]:
```
data.trip_type.value_counts()
```

Out[75]:
```
RoundTrip     49497
OneWay          387
CircleTrip      116
Name: trip_type, dtype: int64
```

In [76]:
```
data.route.value_counts()
```

Out[76]:
```
AKLKUL    2680
PENTPE     924
MELSGN     842
ICNSIN     801
DMKKIX     744
          ... 
LBUTPE       1
CXRMEL       1
DELKBR       1
KOSSYD       1
MRUXIY       1
Name: route, Length: 799, dtype: int64
```

In [77]:
```
data.booking_origin.value_counts()
```

Out[77]:
```
Australia              17872
Malaysia                7174
South Korea             4559
Japan                   3885
China                   3387
                        ...
Panama                     1
Tonga                      1
Tanzania                   1
Bulgaria                   1
Svalbard & Jan Mayen       1
Name: booking_origin, Length: 104, dtype: int64
```

Compare to other countries Australians showed interest to book a ticket.

In [10]:
```
# view the number of duplicates present in our dataset
data.duplicated().sum()
```

Out[10]:
```
719
```

In [11]:
```
data = data.drop_duplicates().reset_index(drop=True)
data.head()
```

Out[11]:

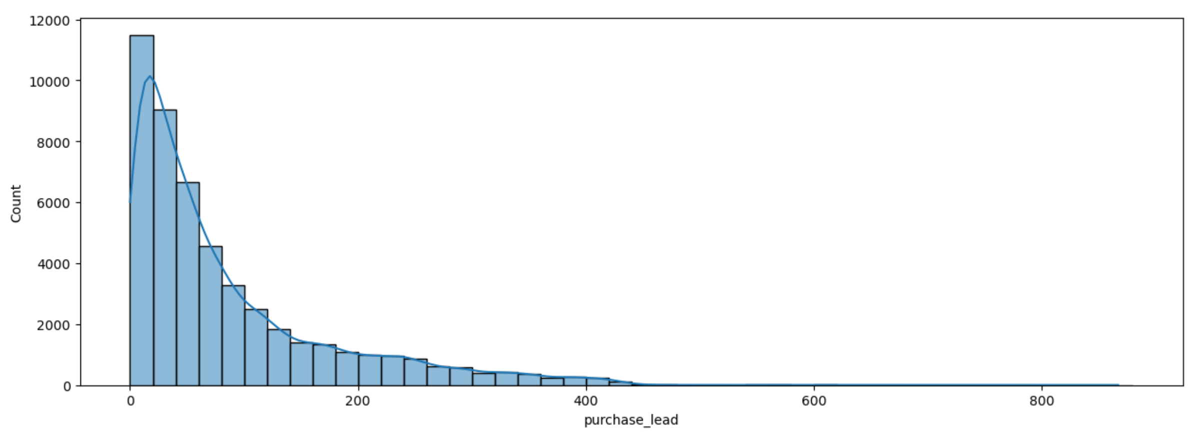
	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	flight_
0	2	Internet	RoundTrip	262	19	7	
1	1	Internet	RoundTrip	112	20	3	
2	2	Internet	RoundTrip	243	22	17	V
3	1	Internet	RoundTrip	96	31	4	
4	2	Internet	RoundTrip	68	22	15	V

In [12]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49281 entries, 0 to 49280
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   num_passengers         49281 non-null  int64  
 1   sales_channel          49281 non-null  object 
 2   trip_type              49281 non-null  object 
 3   purchase_lead          49281 non-null  int64  
 4   length_of_stay         49281 non-null  int64  
 5   flight_hour            49281 non-null  int64  
 6   flight_day             49281 non-null  object 
 7   route                  49281 non-null  object 
 8   booking_origin         49281 non-null  object 
 9   wants_extra_baggage    49281 non-null  int64  
 10  wants_preferred_seat   49281 non-null  int64  
 11  wants_in_flight_meals  49281 non-null  int64  
 12  flight_duration        49281 non-null  float64
 13  booking_complete       49281 non-null  int64  
dtypes: float64(1), int64(8), object(5)
memory usage: 5.3+ MB
```

In [13]:
```
plt.figure(figsize=(15,5))
sns.histplot(data, x="purchase_lead", binwidth=20,kde=True) #kernel Density Fu
```

Out[13]:
```
<AxesSubplot:xlabel='purchase_lead', ylabel='Count'>
```



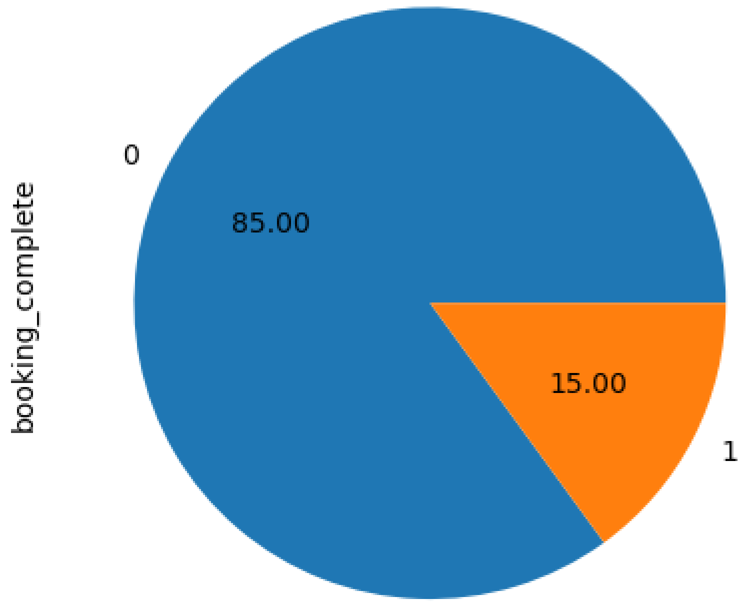
The graph shows most of the people showing interest to book a ticket before their month of journey.

Imbalanced data

In [14]:
```
data.booking_complete.value_counts()
```

Out[14]:
```
0    41890
1     7391
Name: booking_complete, dtype: int64
```

In [15]:
```
data.booking_complete.value_counts().plot.pie(autopct = '%.2f')
```
Out[15]:
```
<AxesSubplot:ylabel='booking_complete'>
```



In [16]:
```
count_not_comp = len(data[data['booking_complete']==0])
count_comp = len(data[data['booking_complete']==1])
Book_not_comp = count_not_comp/(count_not_comp+count_comp)
print("percentage of Booking not Completed is", Book_not_comp*100)
Book_comp = count_comp/(count_not_comp+count_comp)
print("percentage Booking Completed", Book_comp*100)
```

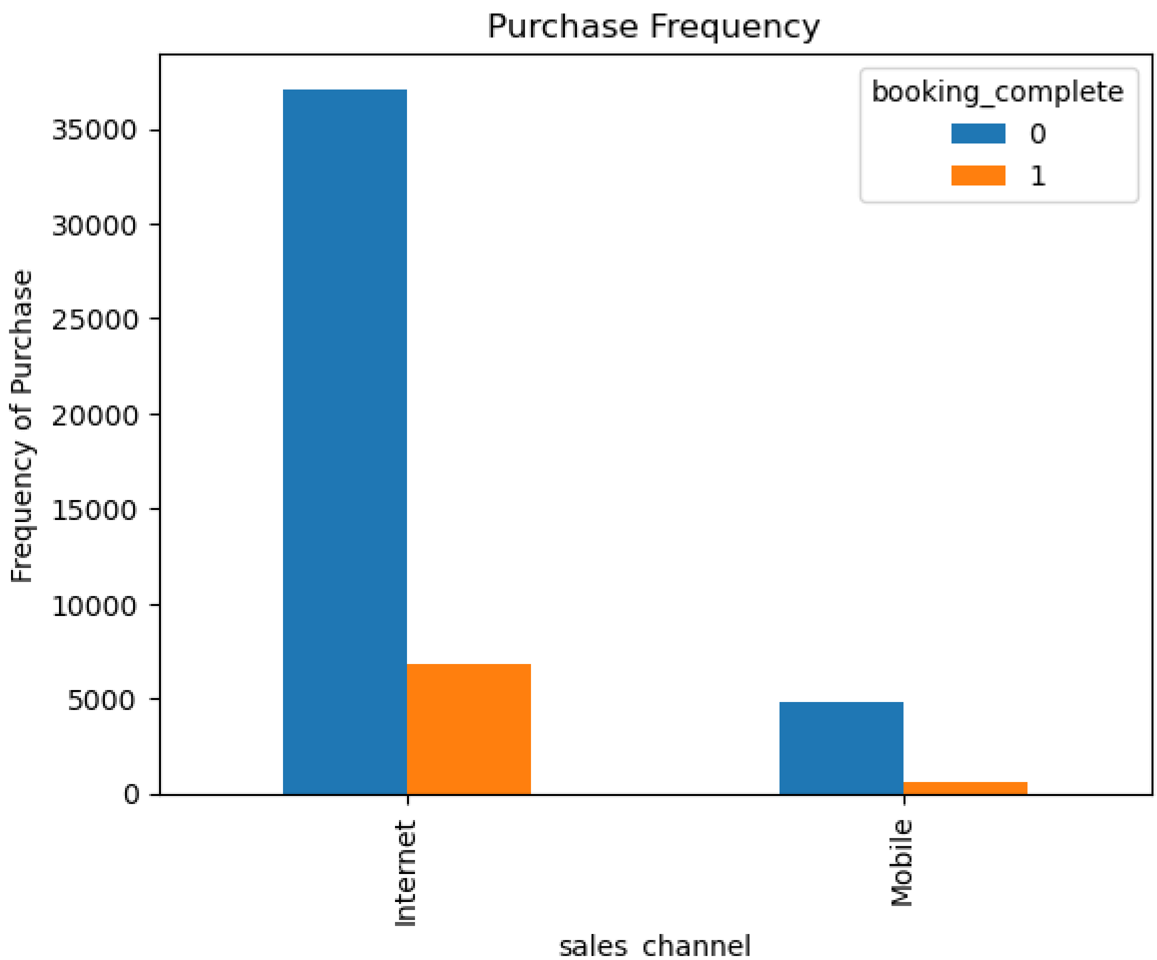
```
percentage of Booking not Completed is 85.00233355654309
percentage Booking Completed 14.997666443456911
```

Our classes are imbalanced, and the ratio of Booking not completed to Booking Completed instances is 85:14.

Before we go ahead to balance the classes, let’s do some more exploration.

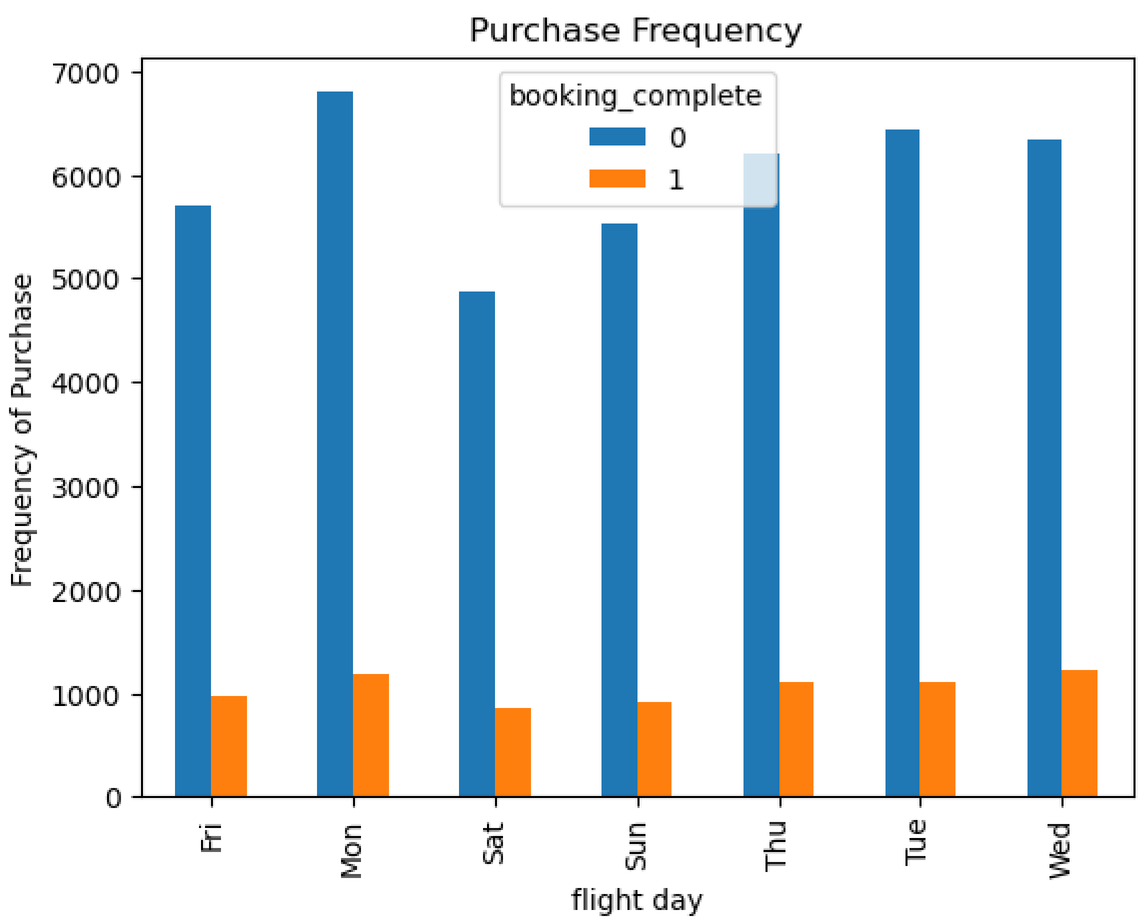
In [17]:
```
%matplotlib inline
pd.crosstab(data.sales_channel,data.booking_complete).plot(kind='bar')
plt.title('Purchase Frequency')
plt.xlabel('sales_channel')
plt.ylabel('Frequency of Purchase')
```

Out[17]:
```
Text(0, 0.5, 'Frequency of Purchase')
```



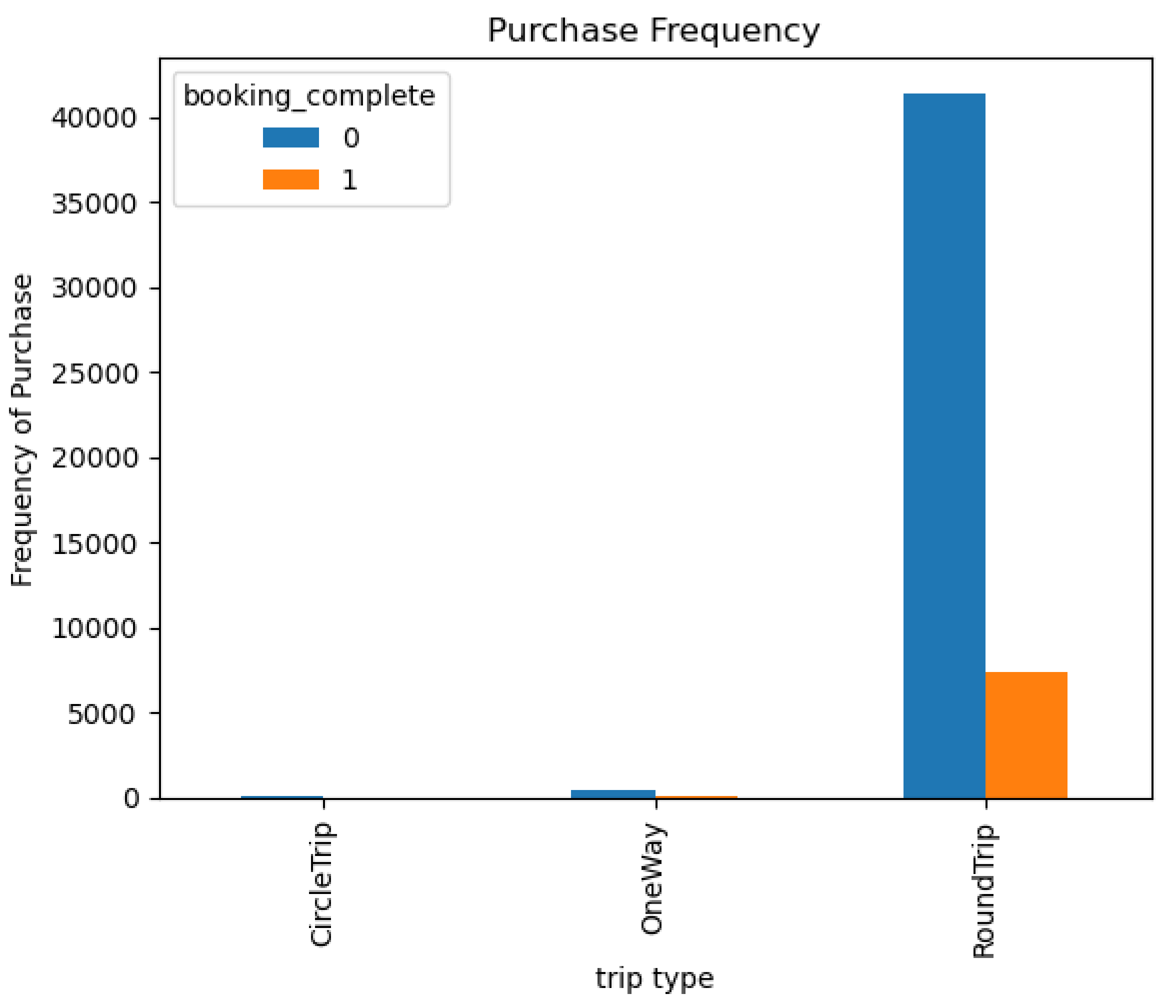
In [18]:
```
pd.crosstab(data.flight_day,data.booking_complete).plot(kind='bar')
plt.title('Purchase Frequency')
plt.xlabel('flight day')
plt.ylabel('Frequency of Purchase')
```

Out[18]:
```
Text(0, 0.5, 'Frequency of Purchase')
```



In [19]:
```python
pd.crosstab(data.trip_type,data.booking_complete).plot(kind='bar')
plt.title('Purchase Frequency')
plt.xlabel('trip type')
plt.ylabel('Frequency of Purchase')
```

Out[19]: Text(0, 0.5, 'Frequency of Purchase')



In [20]:
```python
# Dropping the two fields
data.drop(['route','booking_origin'],axis=1, inplace=True)
```

In [21]:
```python
data.head()
```

Out[21]:

type	purchase_lead	length_of_stay	flight_hour	flight_day	wants_extra_baggage	wants_preferred_
dTrip	262	19	7	Sat	1	
dTrip	112	20	3	Sat	0	
dTrip	243	22	17	Wed	1	
dTrip	96	31	4	Sat	0	
dTrip	68	22	15	Wed	1	

In [22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49281 entries, 0 to 49280
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   num_passengers         49281 non-null  int64  
 1   sales_channel          49281 non-null  object 
 2   trip_type              49281 non-null  object 
 3   purchase_lead          49281 non-null  int64  
 4   length_of_stay         49281 non-null  int64  
 5   flight_hour            49281 non-null  int64  
 6   flight_day             49281 non-null  object 
 7   wants_extra_baggage    49281 non-null  int64  
 8   wants_preferred_seat   49281 non-null  int64  
 9   wants_in_flight_meals  49281 non-null  int64  
 10  flight_duration        49281 non-null  float64
 11  booking_complete       49281 non-null  int64  
dtypes: float64(1), int64(8), object(3)
memory usage: 4.5+ MB
```

In [23]:
```
# creating dummy variables for categorical fields
cat_vars=['sales_channel','trip_type','flight_day']
for var in cat_vars:
    cat_list='var'+'_'+var
    cat_list = pd.get_dummies(data[var], prefix=var)
    data1=data.join(cat_list)
    data=data1
cat_vars=['sales_channel','trip_type','flight_day']
data_vars=data.columns.values.tolist()
to_keep=[i for i in data_vars if i not in cat_vars]
```

In [24]:
```
data_final=data[to_keep]
data_final.columns.values
```

Out[24]:
```
array(['num_passengers', 'purchase_lead', 'length_of_stay', 'flight_hour',
       'wants_extra_baggage', 'wants_preferred_seat',
       'wants_in_flight_meals', 'flight_duration', 'booking_complete',
       'sales_channel_Internet', 'sales_channel_Mobile',
       'trip_type_CircleTrip', 'trip_type_OneWay', 'trip_type_RoundTrip',
       'flight_day_Fri', 'flight_day_Mon', 'flight_day_Sat',
       'flight_day_Sun', 'flight_day_Thu', 'flight_day_Tue',
       'flight_day_Wed'], dtype=object)
```

In [25]:
```
data_final.head()
```

Out[25]:

e_CircleTrip	trip_type_OneWay	trip_type_RoundTrip	flight_day_Fri	flight_day_Mon	flight_day_Sat
0	0	1	0	0	1
0	0	1	0	0	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	0	0

In [26]:
```
data_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49281 entries, 0 to 49280
Data columns (total 21 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   num_passengers         49281 non-null  int64
 1   purchase_lead          49281 non-null  int64
 2   length_of_stay         49281 non-null  int64
 3   flight_hour            49281 non-null  int64
 4   wants_extra_baggage    49281 non-null  int64
 5   wants_preferred_seat   49281 non-null  int64
 6   wants_in_flight_meals  49281 non-null  int64
 7   flight_duration        49281 non-null  float64
 8   booking_complete       49281 non-null  int64
 9   sales_channel_Internet 49281 non-null  uint8
 10  sales_channel_Mobile   49281 non-null  uint8
 11  trip_type_CircleTrip   49281 non-null  uint8
 12  trip_type_OneWay       49281 non-null  uint8
 13  trip_type_RoundTrip    49281 non-null  uint8
 14  flight_day_Fri         49281 non-null  uint8
 15  flight_day_Mon         49281 non-null  uint8
 16  flight_day_Sat         49281 non-null  uint8
 17  flight_day_Sun         49281 non-null  uint8
 18  flight_day_Thu         49281 non-null  uint8
 19  flight_day_Tue         49281 non-null  uint8
 20  flight_day_Wed         49281 non-null  uint8
dtypes: float64(1), int64(8), uint8(12)
memory usage: 3.9 MB
```

Balancing the dataset

In [27]:

```
# With help of SMOTE we can oversample a minority class in our response variab
X = data_final.loc[:, data_final.columns != 'booking_complete']
y = data_final.loc[:, data_final.columns == 'booking_complete']
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
os = SMOTE(random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando
columns = X_train.columns
os_data_X,os_data_y=os.fit_resample(X_train, y_train)
os_data_X = pd.DataFrame(data=os_data_X,columns=columns )
os_data_y= pd.DataFrame(data=os_data_y,columns=['booking_complete'])
# we can Check the numbers of our data
print("length of oversampled data is ",len(os_data_X))
print("Number of person not completed their booking in oversampled data",len(o
print("Number of person completed their booking",len(os_data_y[os_data_y['book
print("Proportion of not purchased data in oversampled data is ",len(os_data_y
print("Proportion of purchased data in oversampled data is ",len(os_data_y[os_
```

```
length of oversampled data is  58694
Number of person not completed their booking in oversampled data 29347
Number of person completed their booking 29347
Proportion of not purchased data in oversampled data is  0.5
Proportion of purchased data in oversampled data is  0.5
```

In [29]:

```
# Recursive feature elimination technique
data_final_vars=data_final.columns.values.tolist()
y=['booking_complete']
X=[i for i in data_final_vars if i not in y]
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
ranfc = RandomForestClassifier()
rfc = RFE(ranfc, n_features_to_select=20)
rfc = rfc.fit(os_data_X, os_data_y.values.ravel())
print(rfc.support_)
print(rfc.ranking_)
```

```
[ True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

we have got all the features value as one which means all the features are important. so, we will consider all the features for our model.

In [30]:
```
cols=['num_passengers','purchase_lead','length_of_stay','flight_hour','wants_e
       'trip_type_CircleTrip', 'trip_type_OneWay', 'trip_type_RoundTrip',
       'flight_day_Fri', 'flight_day_Mon', 'flight_day_Sat',
       'flight_day_Sun', 'flight_day_Thu', 'flight_day_Tue',
       'flight_day_Wed']
X=os_data_X[cols]
y=os_data_y['booking_complete']
```

In [31]:
```
final = X.join(y)
```

In [32]:
```
final.head()
```

Out[32]:

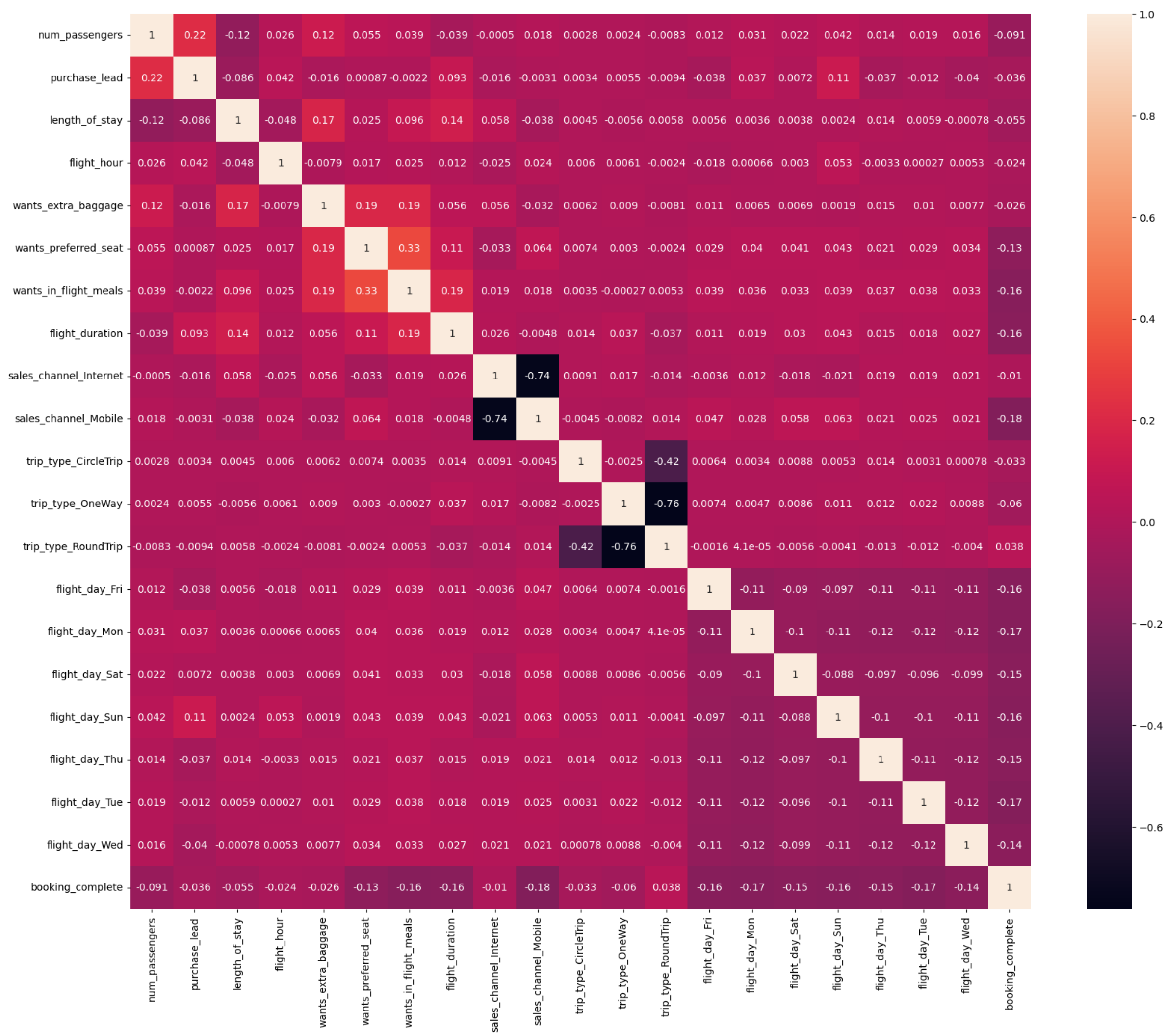
	num_passengers	purchase_lead	length_of_stay	flight_hour	wants_extra_baggage	wants_pref
0	1	3	51	14	0	
1	1	53	5	0	1	
2	1	121	59	4	1	
3	2	57	17	6	0	
4	1	67	18	3	0	

5 rows × 21 columns

```
In [43]: corr = final.corr()

         #plot the heatmap
         plt.figure(figsize=(20,16))
         sns.heatmap(corr, annot = True)
```

Out[43]: <AxesSubplot:>



If the person asking for the preferred seat is more likely ask a flight meals.

Model Creation

```
In [44]: from sklearn.ensemble import RandomForestClassifier
         from sklearn import metrics
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando
         ranfc = RandomForestClassifier()
         ranfc.fit(X_train, y_train)
```

```
Out[44]: RandomForestClassifier()
```

In [45]:
```
y_pred = ranfc.predict(X_test)
print('Accuracy of Random Forest classifier on test set: {:.2f}'.format(ranfc.s
```

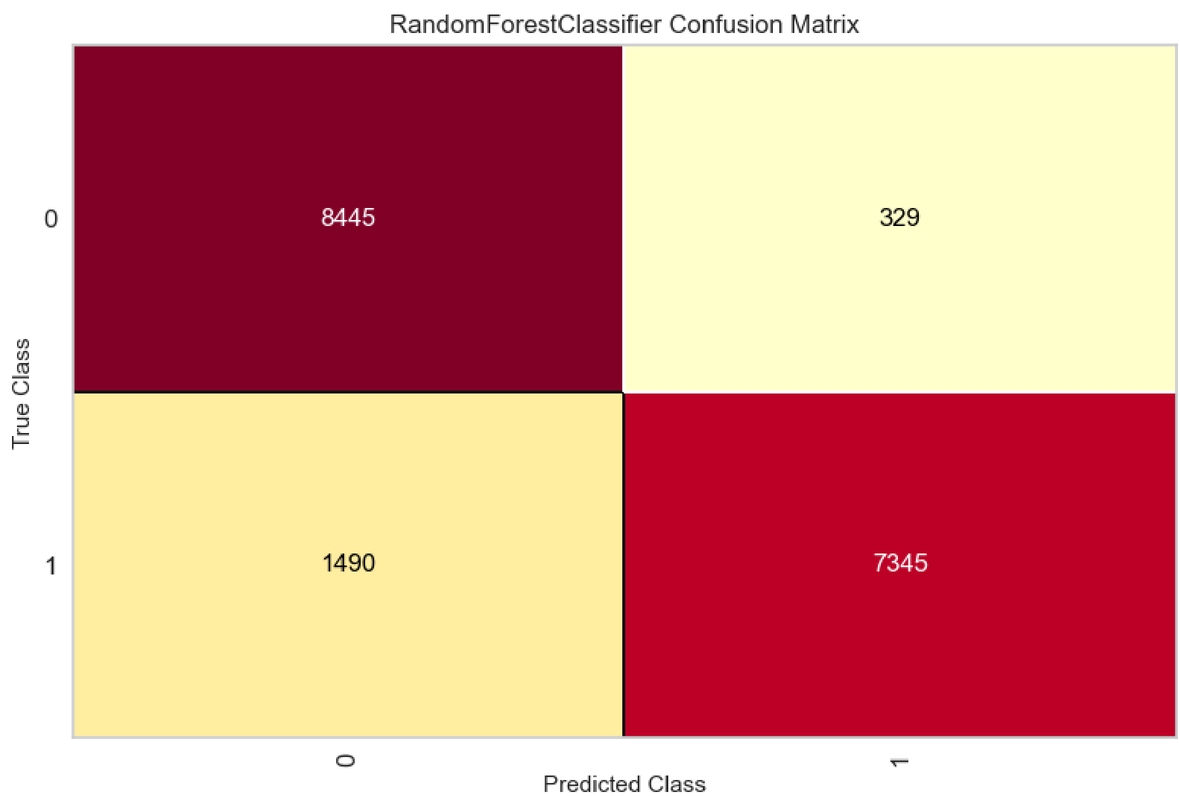
```
Accuracy of Random Forest classifier on test set: 0.90
```

In [46]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[8445  329]
 [1490 7345]]
```

In [47]:
```
from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(
    ranfc, classes=[0,1],
    percent=False)
cm.fit(X_train, y_train)
cm.score(X_test, y_test)
cm.show();
```

```
C:\Users\yukym\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning:
X does not have valid feature names, but RandomForestClassifier was fitted wi
th feature names
  warnings.warn(
```



In [48]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.96      0.90      8774
           1       0.96      0.83      0.89      8835

    accuracy                           0.90     17609
   macro avg       0.90      0.90      0.90     17609
weighted avg       0.90      0.90      0.90     17609
```

HyperParameter tuning

In [49]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
param_dist = {'n_estimators': randint(50,500),
              'max_depth': randint(1,20)}

# Create a random forest classifier
rf = RandomForestClassifier()

# Use random search to find the best hyperparameters
rand_search = RandomizedSearchCV(rf,
                                 param_distributions = param_dist,
                                 n_iter=5,
                                 cv=5)

# Fit the random search object to the data
rand_search.fit(X_train, y_train)
```

Out[49]:
```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(), n_iter=5,
                   param_distributions={'max_depth': <scipy.stats._distn_infr
astructure.rv_discrete_frozen object at 0x0000025C4E598520>,
                                        'n_estimators': <scipy.stats._distn_i
nfrastructure.rv_discrete_frozen object at 0x0000025C4E52F8E0>})
```

In [50]:
```
# Create a variable for the best model
best_rf = rand_search.best_estimator_

# Print the best hyperparameters
print('Best hyperparameters:',  rand_search.best_params_)
```

```
Best hyperparameters: {'max_depth': 17, 'n_estimators': 265}
```

In [59]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando
ranfc = RandomForestClassifier(max_depth= 26, n_estimators= 400)
ranfc.fit(X_train, y_train)
```

Out[59]:
```
RandomForestClassifier(max_depth=26, n_estimators=400)
```

In [60]:
```
y_pred = ranfc.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(
```

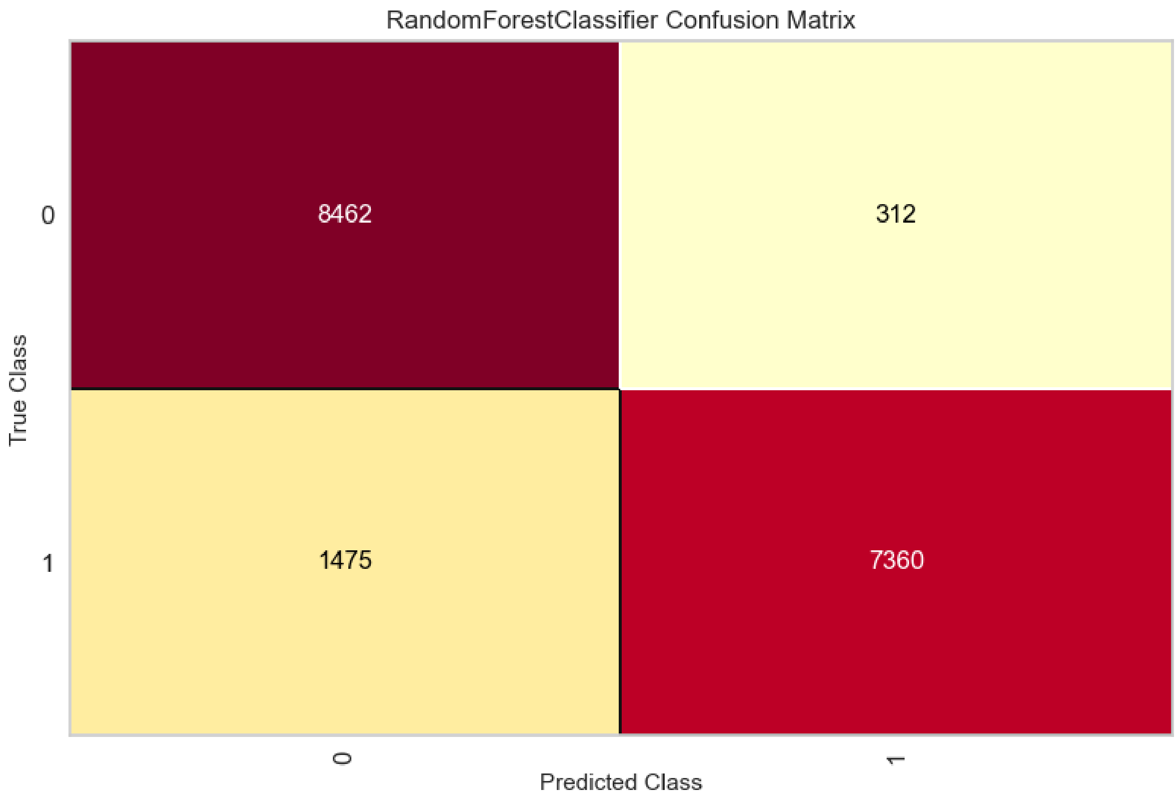
```
Accuracy of logistic regression classifier on test set: 0.90
```

In [61]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[8462  312]
 [1475 7360]]
```

In [62]:
```
from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(
    ranfc, classes=[0,1],
    percent=False)
cm.fit(X_train, y_train)
cm.score(X_test, y_test)
cm.show();
```

```
C:\Users\yukym\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning:
X does not have valid feature names, but RandomForestClassifier was fitted wi
th feature names
  warnings.warn(
```



In [63]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.96      0.90      8774
           1       0.96      0.83      0.89      8835

    accuracy                           0.90     17609
   macro avg       0.91      0.90      0.90     17609
weighted avg       0.91      0.90      0.90     17609
```

From the above report we can see that after, Hyperparameter tuning precision percentage is increased one percent.

Atlast, visualizing which factors impact more towards the response variable.

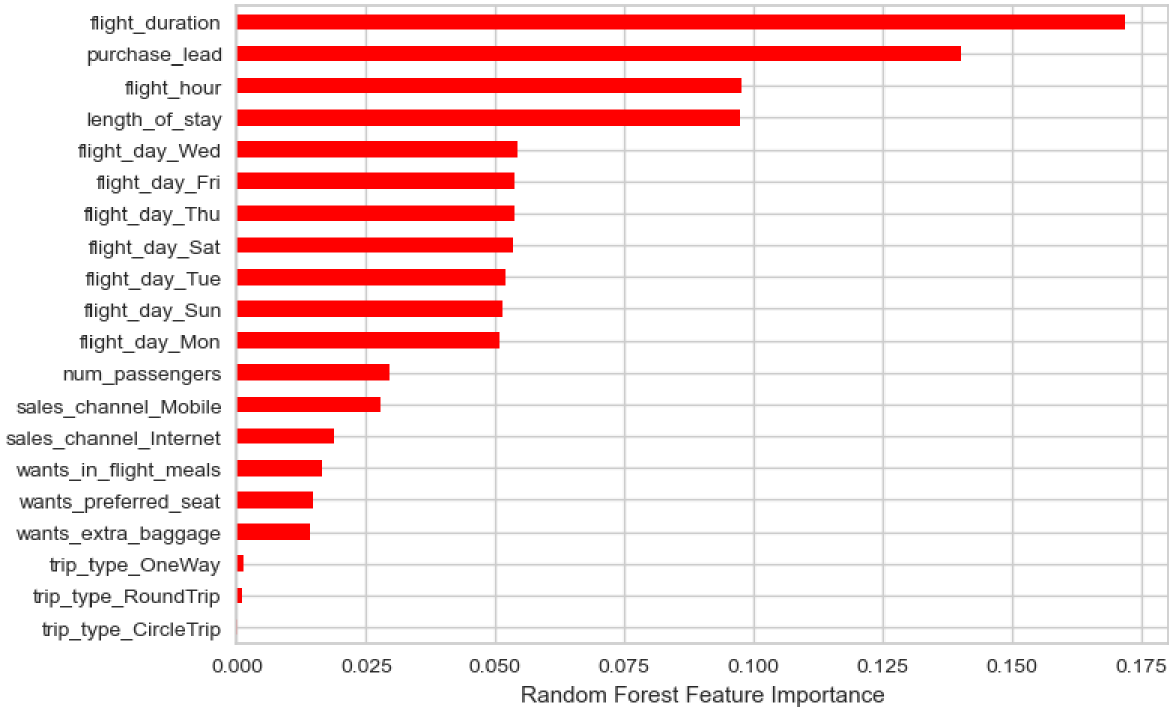
In [64]:
```
#finding which variable have more impact to the target variable
importance = ranfc.feature_importances_
columns = X_train.columns
```

In [65]:
```
rfc_cof = pd.Series(importance, columns)
rfc_cof
```

Out[65]:
```
num_passengers           0.029588
purchase_lead            0.140098
length_of_stay           0.097415
flight_hour              0.097650
wants_extra_baggage      0.014093
wants_preferred_seat     0.014734
wants_in_flight_meals    0.016549
flight_duration          0.171814
sales_channel_Internet   0.018886
sales_channel_Mobile     0.027752
trip_type_CircleTrip     0.000286
trip_type_OneWay         0.001315
trip_type_RoundTrip      0.001127
flight_day_Fri           0.053633
flight_day_Mon           0.050670
flight_day_Sat           0.053271
flight_day_Sun           0.051421
flight_day_Thu           0.053556
flight_day_Tue           0.051962
flight_day_Wed           0.054182
dtype: float64
```

In [67]:
```
%matplotlib inline
rfc_cof.sort_values().plot.barh(color='red')
plt.xlabel("Random Forest Feature Importance")
```

Out[67]: Text(0.5, 0, 'Random Forest Feature Importance')



We can conclude that flight duration contribute more towards customer booking .

In [ ]: