INTERNSHIP

```
import numpy as np
import pandas as pd

file =
pd.read_csv("C:/Users/srksa/Downloads/customer_booking.csv",encoding='
latin-1')

file.head(5)
```

```
   num_passengers sales_channel  trip_type  purchase_lead
length_of_stay  \
0               2      Internet  RoundTrip            262
19
1               1      Internet  RoundTrip            112
20
2               2      Internet  RoundTrip            243
22
3               1      Internet  RoundTrip             96
31
4               2      Internet  RoundTrip             68
22

   flight_hour flight_day   route booking_origin  wants_extra_baggage
\
0            7         Sat  AKLDEL    New Zealand                     1

1            3         Sat  AKLDEL    New Zealand                     0

2           17         Wed  AKLDEL          India                     1

3            4         Sat  AKLDEL    New Zealand                     0

4           15         Wed  AKLDEL          India                     1


   wants_preferred_seat  wants_in_flight_meals  flight_duration  \
0                     0                      0             5.52
1                     0                      0             5.52
2                     1                      0             5.52
3                     0                      1             5.52
4                     0                      1             5.52

   booking_complete
0                 0
1                 0
2                 0
3                 0
4                 0
```

```
file.tail(5)
```

```
       num_passengers sales_channel  trip_type  purchase_lead
length_of_stay  \
49995               2      Internet  RoundTrip             27
6
49996               1      Internet  RoundTrip            111
6
49997               1      Internet  RoundTrip             24
6
49998               1      Internet  RoundTrip             15
6
49999               1      Internet  RoundTrip             19
6

       flight_hour flight_day   route booking_origin
wants_extra_baggage  \
49995             9         Sat  PERPNH      Australia
1
49996             4         Sun  PERPNH      Australia
0
49997            22         Sat  PERPNH      Australia
0
49998            11         Mon  PERPNH      Australia
1
49999            10         Thu  PERPNH      Australia
0

       wants_preferred_seat  wants_in_flight_meals  flight_duration  \
49995                     0                      1             5.62
49996                     0                      0             5.62
49997                     0                      1             5.62
49998                     0                      1             5.62
49999                     1                      0             5.62

       booking_complete
49995                 0
49996                 0
49997                 0
49998                 0
49999                 0
```

```
file.isnull().sum()
```

```
num_passengers           0
sales_channel            0
trip_type                0
purchase_lead            0
length_of_stay           0
flight_hour              0
```

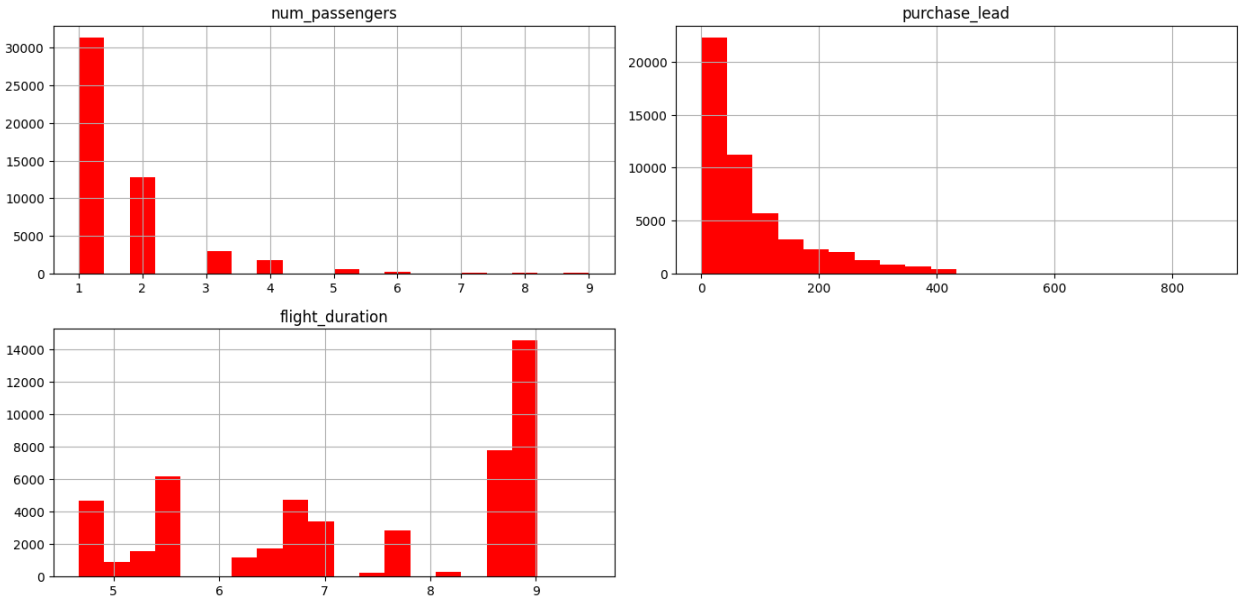
```
flight_day                0
route                     0
booking_origin            0
wants_extra_baggage       0
wants_preferred_seat      0
wants_in_flight_meals     0
flight_duration           0
booking_complete          0
dtype: int64
```

Data Exploration and Basic Analysis

```
numerical_features =
['num_passengers','purchase_lead','flight_duration']

num_sts = file[numerical_features].describe()



modes = file[numerical_features].mode().iloc[0]


print(num_sts)

print("\nmode\n",modes)
```

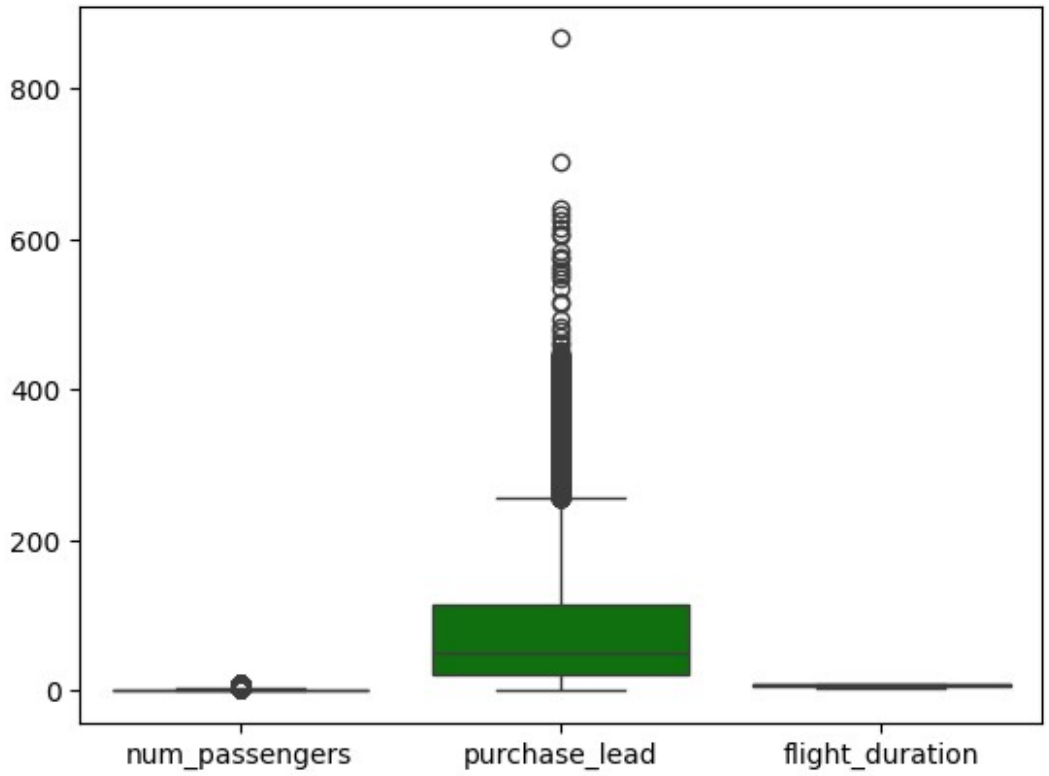
```
       num_passengers  purchase_lead  flight_duration
count    50000.000000   50000.000000     50000.000000
mean         1.591240      84.940480         7.277561
std          1.020165      90.451378         1.496863
min          1.000000       0.000000         4.670000
25%          1.000000      21.000000         5.620000
50%          1.000000      51.000000         7.570000
75%          2.000000     115.000000         8.830000
max          9.000000     867.000000         9.500000

mode
 num_passengers     1.00
purchase_lead      1.00
flight_duration    8.83
Name: 0, dtype: float64
```

```
import matplotlib.pyplot as plt

file[numerical_features].hist(bins=20, figsize=(14, 10), layout=(3,
2),color="red")
plt.tight_layout()
plt.show()
```




```
import seaborn as sns

sns.boxplot(data = file[numerical_features],color="green")
```

```
<Axes: >
```



```
sls = file['sales_channel'].value_counts()
sls
```

```
sales_channel
Internet    44382
Mobile       5618
Name: count, dtype: int64
```

```
trp = file['trip_type'].value_counts()
trp
```

```
trip_type
RoundTrip     49497
OneWay          387
CircleTrip      116
Name: count, dtype: int64
```

```
bk = file['booking_origin'].value_counts()
bk
```

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

DemandPrediction

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error , mean_absolute_error

from math import sqrt

x = file[['purchase_lead','length_of_stay','flight_duration']]

y = file["num_passengers"]


x_train,x_text,y_train,y_test = train_test_split(x,y,
test_size=0.2,random_state= 42)

model = LinearRegression()
```

```
model.fit(x_train,y_train)

y_pred = model.predict(x_text)

mae = mean_absolute_error(y_test,y_pred)
mse = mean_squared_error(y_test,y_pred)

print("MAE=",mae)
print("RMSE=",mse)
```

```
MAE= 0.6991936588385864
RMSE= 1.0178093132829988
```

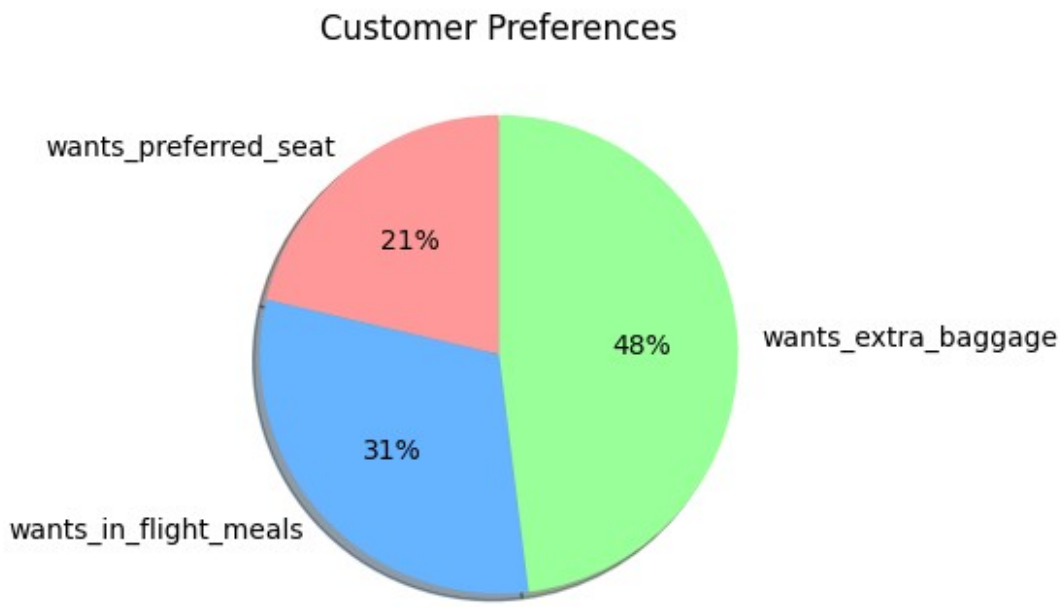
```
def predict_passengers(model):
    print("Enter the values for the following features:")
    purchase_lead_time = float(input("Purchase Lead Time: "))
    length_of_stay = int(input("Length of Stay: "))
    flight_duration = float(input("Flight Duration: "))


    input_data = pd.DataFrame({
        'purchase_lead': [purchase_lead_time],
        'length_of_stay': [length_of_stay],
        'flight_duration': [flight_duration]
    })


    prediction = model.predict(input_data)

    print("Predicted number of passengers:", prediction[0])


predict_passengers(model)
```

```
Enter the values for the following features:
Predicted number of passengers: 1.7014095096120279
```

Customer Preference Analysis

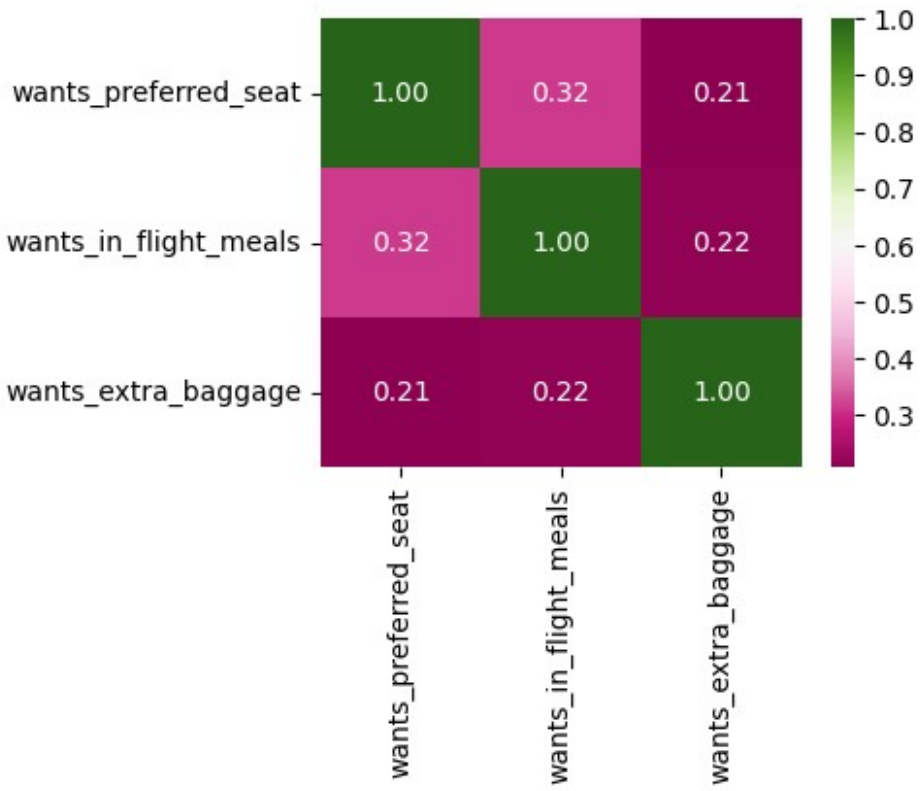
```
choice = file[['wants_preferred_seat', 'wants_in_flight_meals',
'wants_extra_baggage']]


total_counts = choice.sum()


colors = ['#FF9999', '#66B3FF', '#99FF99']
```

```
plt.figure(figsize=(6, 4))
plt.pie(total_counts, labels=total_counts.index, autopct='%1.0f%%',
startangle=90, colors=colors, shadow=True)

plt.title('Customer Preferences')
plt.show()
```




```
choice = pd.get_dummies(choice)

correlation_matrix = choice.corr()

plt.figure(figsize=(4,3))
sns.heatmap(correlation_matrix,annot=True,fmt="0.2f",cmap="PiYG")
```

```
<Axes: >
```



Churn Prediction

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import
f1_score,recall_score,precision_score,accuracy_score

x = file[['purchase_lead','length_of_stay','flight_duration']]

y = file["booking_complete"]

x_train,x_text , y_train,y_test = train_test_split(x,y, test_size=0.2,
random_state= 42)




print("DecisionTree Classifier")
dtc_model = DecisionTreeClassifier()
dtc_model.fit(x_train,y_train)

dtc_pred = dtc_model.predict(x_text)
```

```
acc = accuracy_score(y_test,dtc_pred)

rec = recall_score(y_test,dtc_pred)
f1 = f1_score(y_test, dtc_pred)

dtc_precision_score = precision_score(y_test,dtc_pred)

print("Accuracy Score: ",acc)
print("Recall Score: ", rec)
print("F1 Score : ", f1)
print("Precision Score", dtc_precision_score)
```

```
DecisionTree Classifier
Accuracy Score:  0.7994
Recall Score:  0.14391891891891892
F1 Score :  0.17516447368421054
Precision Score 0.22373949579831934
```

```
print("Random Forest Classifier")
rfc_model = RandomForestClassifier()
rfc_model.fit(x_train,y_train)

rfc_pred = rfc_model.predict(x_text)

r_acc = accuracy_score(y_test,rfc_pred)

r_rec = recall_score(y_test,rfc_pred)
r_f1 = f1_score(y_test, rfc_pred)

r_precision_score = precision_score(y_test,dtc_pred)

print("Accuracy Score: ",r_acc)
print("Recall Score: ", r_rec)
print("F1 Score : ", r_f1)
print("Precision Score", r_precision_score)
```

```
Random Forest Classifier
Accuracy Score:  0.815
Recall Score:  0.125
F1 Score :  0.16666666666666666
Precision Score 0.22373949579831934
```

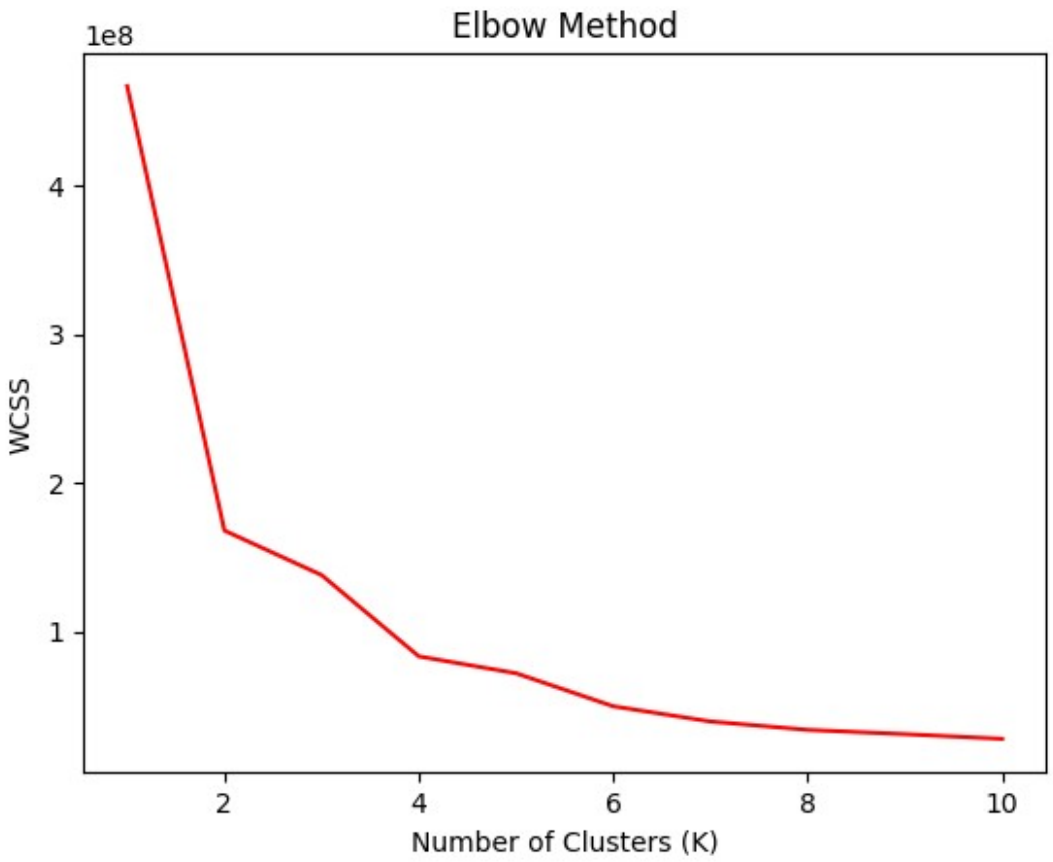
```
def predict_passengers(rfc_model):
    print("Enter the values for the following features:")
    purchase_lead_time = float(input("Purchase Lead Time: "))
    length_of_stay = int(input("Length of Stay: "))
    flight_duration = float(input("Flight Duration: "))


    input_data = pd.DataFrame({
        'purchase_lead': [purchase_lead_time],
```

```
        'length_of_stay': [length_of_stay],
        'flight_duration': [flight_duration]
    })


    Classification = rfc_model.predict(input_data)

    print("TASK:", Classification[0])


predict_passengers(rfc_model)
```

```
Enter the values for the following features:
TASK: 0
```

segment Analysis

```
from sklearn.cluster import KMeans



X = file[['purchase_lead', 'length_of_stay', 'flight_duration']]


wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss,color="red")
plt.xlabel('Number of Clusters (K)')
plt.ylabel('WCSS')
plt.title('Elbow Method')
plt.show()


kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)


file['cluster'] = kmeans.labels_
```



Time Series Analysis

```
df =
pd.read_csv("C:/Users/srksa/Downloads/customer_booking.csv",encoding='
latin-1')

dates = pd.date_range(start='2023-01-01', end='2024-12-31')



repeated_dates = pd.Series(dates).repeat((len(df) // len(dates)) + 1)
[:len(df)]


df['date'] = repeated_dates.values



df['booking_complete'] = df['booking_complete'].astype(int)

df.set_index('date', inplace=True)
```

```
df


            num_passengers sales_channel  trip_type  purchase_lead  \
date
2023-01-01               2      Internet  RoundTrip            262
2023-01-01               1      Internet  RoundTrip            112
2023-01-01               2      Internet  RoundTrip            243
2023-01-01               1      Internet  RoundTrip             96
2023-01-01               2      Internet  RoundTrip             68
...                    ...           ...        ...            ...
2024-12-25               2      Internet  RoundTrip             27
2024-12-25               1      Internet  RoundTrip            111
2024-12-25               1      Internet  RoundTrip             24
2024-12-25               1      Internet  RoundTrip             15
2024-12-25               1      Internet  RoundTrip             19

            length_of_stay  flight_hour flight_day   route
booking_origin  \
date

2023-01-01              19            7        Sat  AKLDEL    New
Zealand
2023-01-01              20            3        Sat  AKLDEL    New
Zealand
2023-01-01              22           17        Wed  AKLDEL
India
2023-01-01              31            4        Sat  AKLDEL    New
Zealand
2023-01-01              22           15        Wed  AKLDEL
India
...                    ...          ...        ...     ...
...
2024-12-25               6            9        Sat  PERPNH
Australia
2024-12-25               6            4        Sun  PERPNH
Australia
2024-12-25               6           22        Sat  PERPNH
Australia
2024-12-25               6           11        Mon  PERPNH
Australia
2024-12-25               6           10        Thu  PERPNH
Australia

            wants_extra_baggage  wants_preferred_seat
wants_in_flight_meals  \
date

2023-01-01                    1                     0
0
```

```
2023-01-01                     0                      0
0
2023-01-01                     1                      1
0
2023-01-01                     0                      0
1
2023-01-01                     1                      0
1
...                          ...                    ...
...
2024-12-25                     1                      0
1
2024-12-25                     0                      0
0
2024-12-25                     0                      0
1
2024-12-25                     1                      0
1
2024-12-25                     0                      1
0

             flight_duration  booking_complete
date
2023-01-01              5.52                 0
2023-01-01              5.52                 0
2023-01-01              5.52                 0
2023-01-01              5.52                 0
2023-01-01              5.52                 0
...                      ...               ...
2024-12-25              5.62                 0
2024-12-25              5.62                 0
2024-12-25              5.62                 0
2024-12-25              5.62                 0
2024-12-25              5.62                 0

[50000 rows x 14 columns]
```

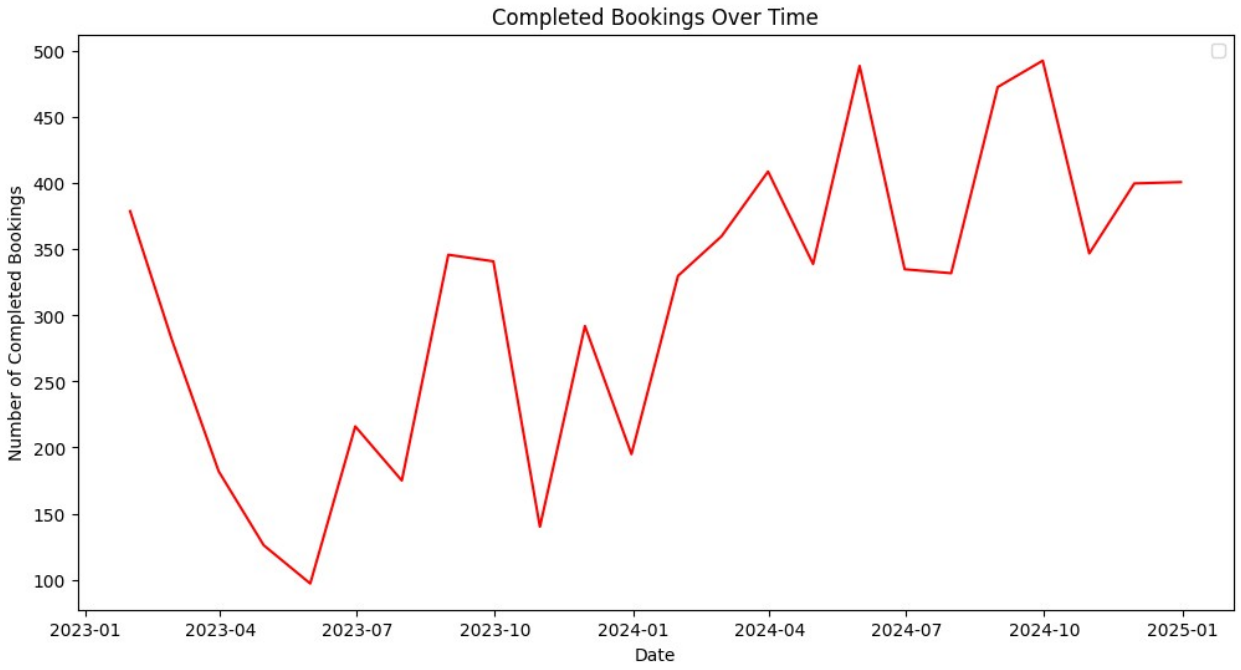
```
monthly_booking_complete = df.resample('M').sum()['booking_complete']
monthly_booking_complete
```

```
C:\Users\srksa\AppData\Local\Temp\ipykernel_22188\96724248.py:1:
FutureWarning: 'M' is deprecated and will be removed in a future
version, please use 'ME' instead.
  monthly_booking_complete = df.resample('M').sum()
['booking_complete']

date
2023-01-31    379
2023-02-28    281
2023-03-31    182
```

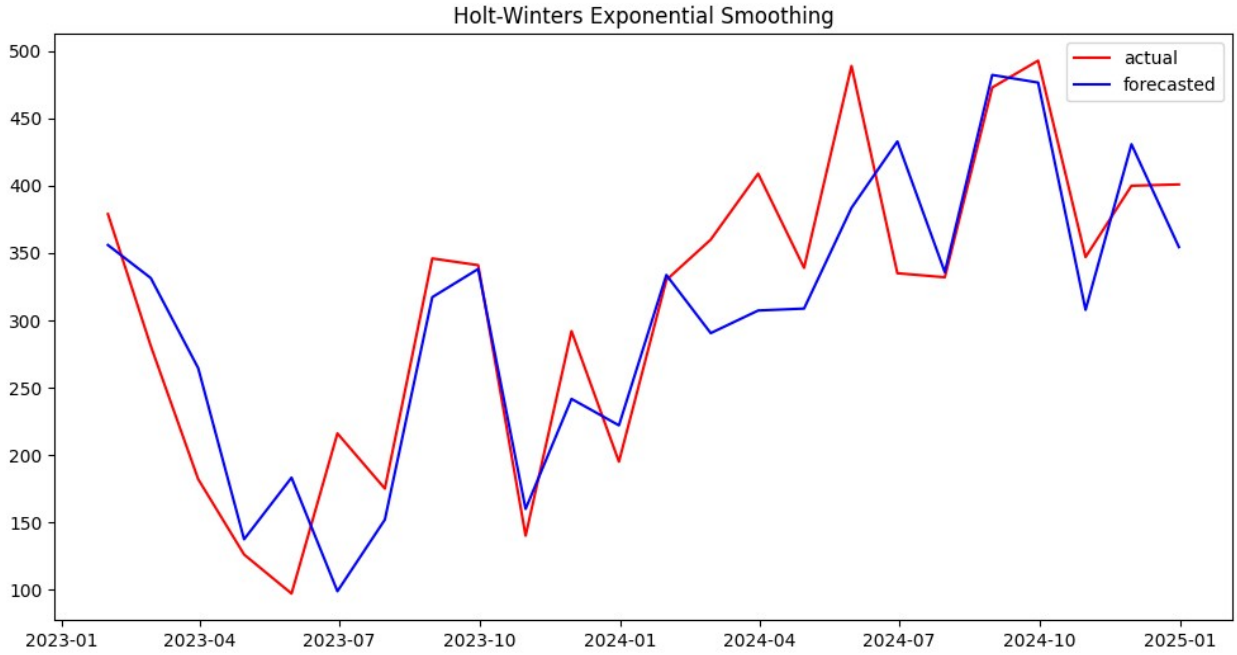
```
2023-04-30    126
2023-05-31     97
2023-06-30    216
2023-07-31    175
2023-08-31    346
2023-09-30    341
2023-10-31    140
2023-11-30    292
2023-12-31    195
2024-01-31    330
2024-02-29    360
2024-03-31    409
2024-04-30    339
2024-05-31    489
2024-06-30    335
2024-07-31    332
2024-08-31    473
2024-09-30    493
2024-10-31    347
2024-11-30    400
2024-12-31    401
Freq: ME, Name: booking_complete, dtype: int32
```

```
plt.figure(figsize=(12,6))

plt.plot(monthly_booking_complete,color="red")

plt.title('Completed Bookings Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Completed Bookings')
plt.legend()
plt.show()
```

```
C:\Users\srksa\AppData\Local\Temp\ipykernel_22188\3956998242.py:8:
UserWarning: No artists with labels found to put in legend.  Note that
artists whose label start with an underscore are ignored when legend()
is called with no argument.
  plt.legend()
```




```
from statsmodels.tsa.holtwinters import ExponentialSmoothing


model = ExponentialSmoothing(monthly_booking_complete,
seasonal='additive', seasonal_periods=12)
fit = model.fit()


plt.figure(figsize=(12, 6))
plt.plot(monthly_booking_complete, label='actual',color="red")
plt.plot(fit.fittedvalues, label='forecasted',color="blue")
plt.title('Holt-Winters Exponential Smoothing')
plt.legend()
plt.show()
```

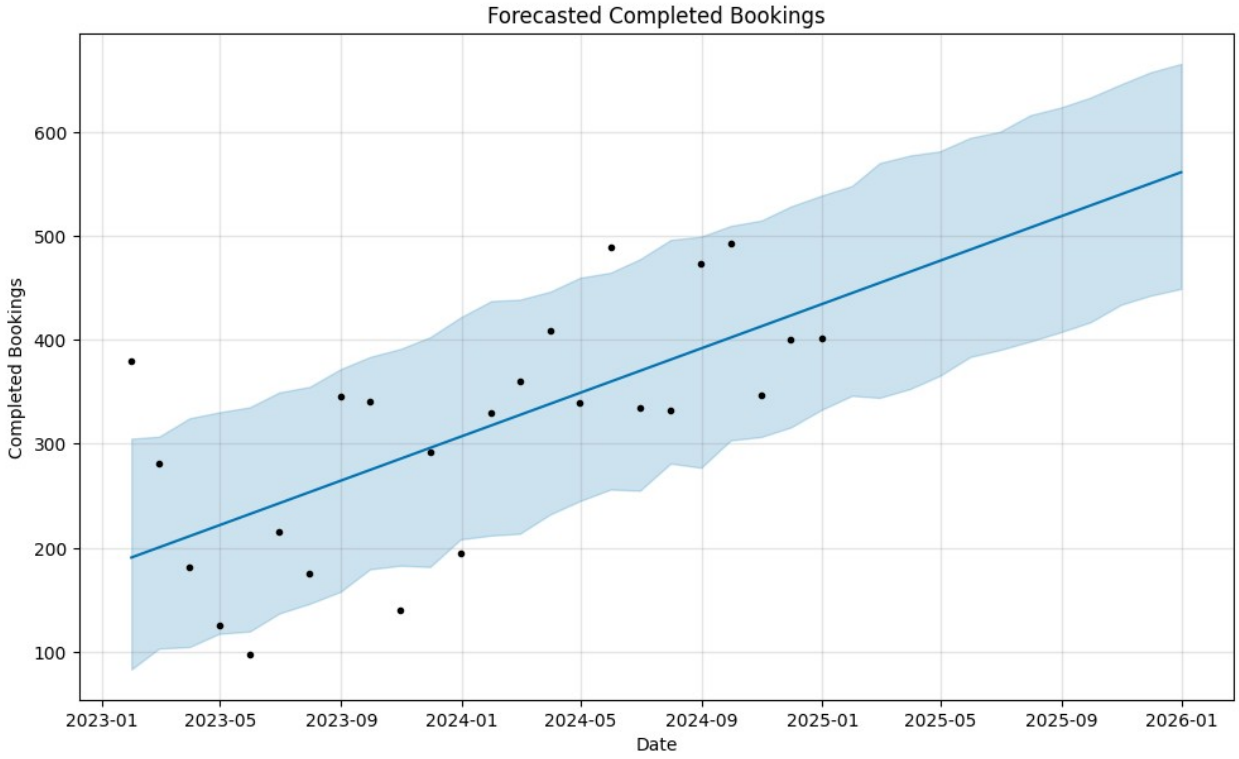



```
from prophet import Prophet

from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

prophet_df =
monthly_booking_complete.reset_index().rename(columns={'date': 'ds',
"booking_complete": "y"})

fr_model = Prophet()
fr_model.fit(prophet_df)

future_dates = fr_model.make_future_dataframe(periods=12, freq='M')

forecast = fr_model.predict(future_dates)



train = prophet_df.iloc[:-12]
test = prophet_df.iloc[-12:]

test_predictions = forecast[-12:]['yhat']

mae = mean_absolute_error(test['y'], test_predictions)
rmse = np.sqrt(mean_squared_error(test['y'], test_predictions))

print('Mean Absolute Error:', mae)
print('Root Mean Squared Error:', rmse)
```

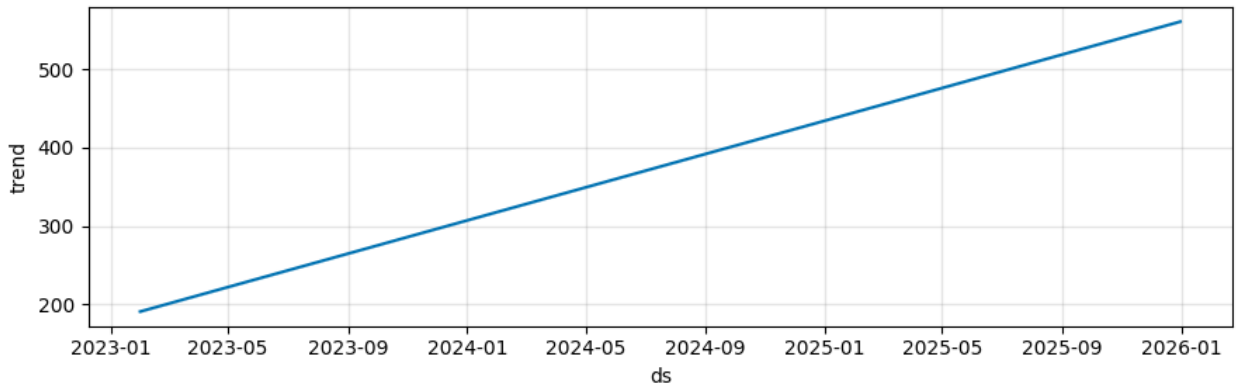
```
fr_model.plot(forecast)
plt.title('Forecasted Completed Bookings')
plt.xlabel('Date')
plt.ylabel('Completed Bookings')
plt.show()

# Plot components
fr_model.plot_components(forecast)
plt.show()
```

```
16:11:31 - cmdstanpy - INFO - Chain [1] start processing
16:11:32 - cmdstanpy - INFO - Chain [1] done processing
C:\Users\srksa\AppData\Roaming\Python\Python312\site-packages\prophet\
forecaster.py:1854: FutureWarning: 'M' is deprecated and will be
removed in a future version, please use 'ME' instead.
  dates = pd.date_range(
C:\Users\srksa\AppData\Roaming\Python\Python312\site-packages\prophet\
plot.py:72: FutureWarning: The behavior of
DatetimeProperties.to_pydatetime is deprecated, in a future version
this will return a Series containing python datetime objects instead
of an ndarray. To retain the old behavior, call `np.array` on the
result
  fcst_t = fcst['ds'].dt.to_pydatetime()
C:\Users\srksa\AppData\Roaming\Python\Python312\site-packages\prophet\
plot.py:73: FutureWarning: The behavior of
DatetimeProperties.to_pydatetime is deprecated, in a future version
this will return a Series containing python datetime objects instead
of an ndarray. To retain the old behavior, call `np.array` on the
result
  ax.plot(m.history['ds'].dt.to_pydatetime(), m.history['y'], 'k.',
```

```
Mean Absolute Error: 110.64640191417743
Root Mean Squared Error: 125.8244924089689
```




```
C:\Users\srksa\AppData\Roaming\Python\Python312\site-packages\prophet\plot.py:228: FutureWarning: The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To retain the old behavior, call `np.array` on the result
  fcst_t = fcst['ds'].dt.to_pydatetime()
```



Feature Importance for Regression

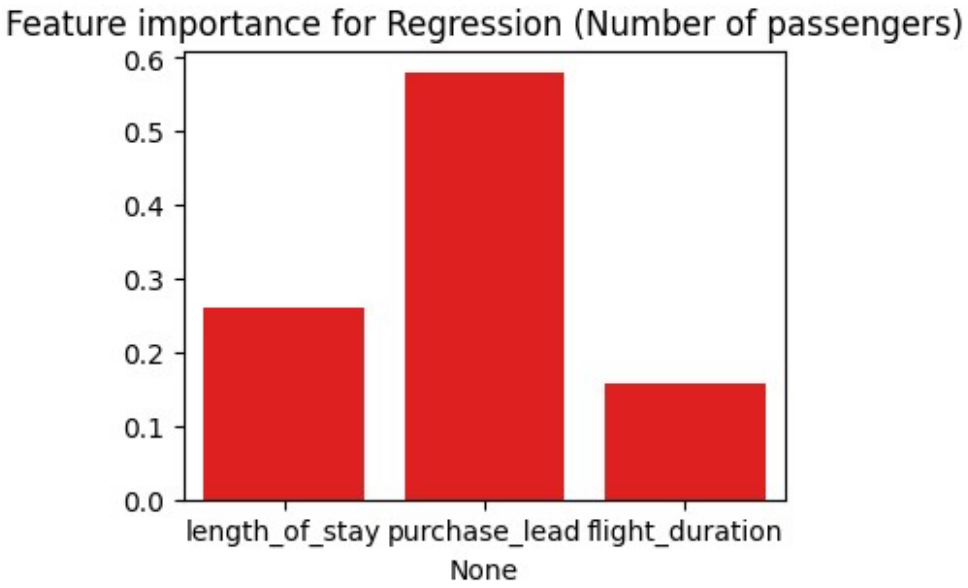
Feature Importance for Regression (Number of Passengers)

```
features = ['length_of_stay','purchase_lead','flight_duration']
target = ["num_passengers"]

x = file[features]
y =file[target]

from sklearn.tree import DecisionTreeRegressor

x_train,x_text,y_train,y_test = train_test_split(x,y, test_size= 0.2,
random_state=42 )

DTR_model = DecisionTreeRegressor(random_state=42)
#??????????????????
DTR_model.fit(x_train,y_train)
```

```
DecisionTreeRegressor(random_state=42)
```

Feature importance

```
import matplotlib.pyplot as plt

import seaborn as sns

feature_importance = DTR_model.feature_importances_
feature_names = x.columns

plt.figure(figsize=(4,3))
sns.barplot(y = feature_importance, x = feature_names,color="red")
plt.title("Feature importance for Regression (Number of passengers)")
plt.show()
```

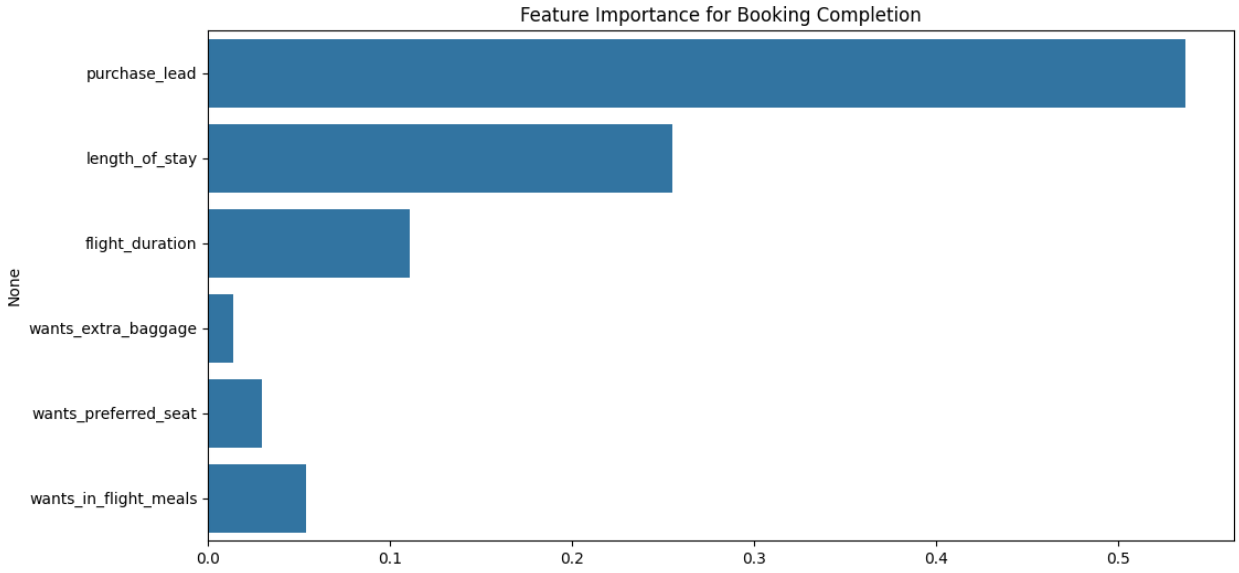


Feature Importance for Classification (Booking Completion)

```
features = ['purchase_lead', 'length_of_stay', 'flight_duration',
'wants_extra_baggage', 'wants_preferred_seat',
'wants_in_flight_meals']
target = 'booking_complete'

X = df[features]
y = df[target]

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier


X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)


DTC_model = DecisionTreeClassifier(random_state=42)
DTC_model.fit(X_train, y_train)
```

```
DecisionTreeClassifier(random_state=42)
```

```
feature_importance = DTC_model.feature_importances_
feature_names = X.columns

# Plot feature importances
plt.figure(figsize=(12, 6))
sns.barplot(x=feature_importance, y=feature_names)
plt.title('Feature Importance for Booking Completion')
plt.show()
```

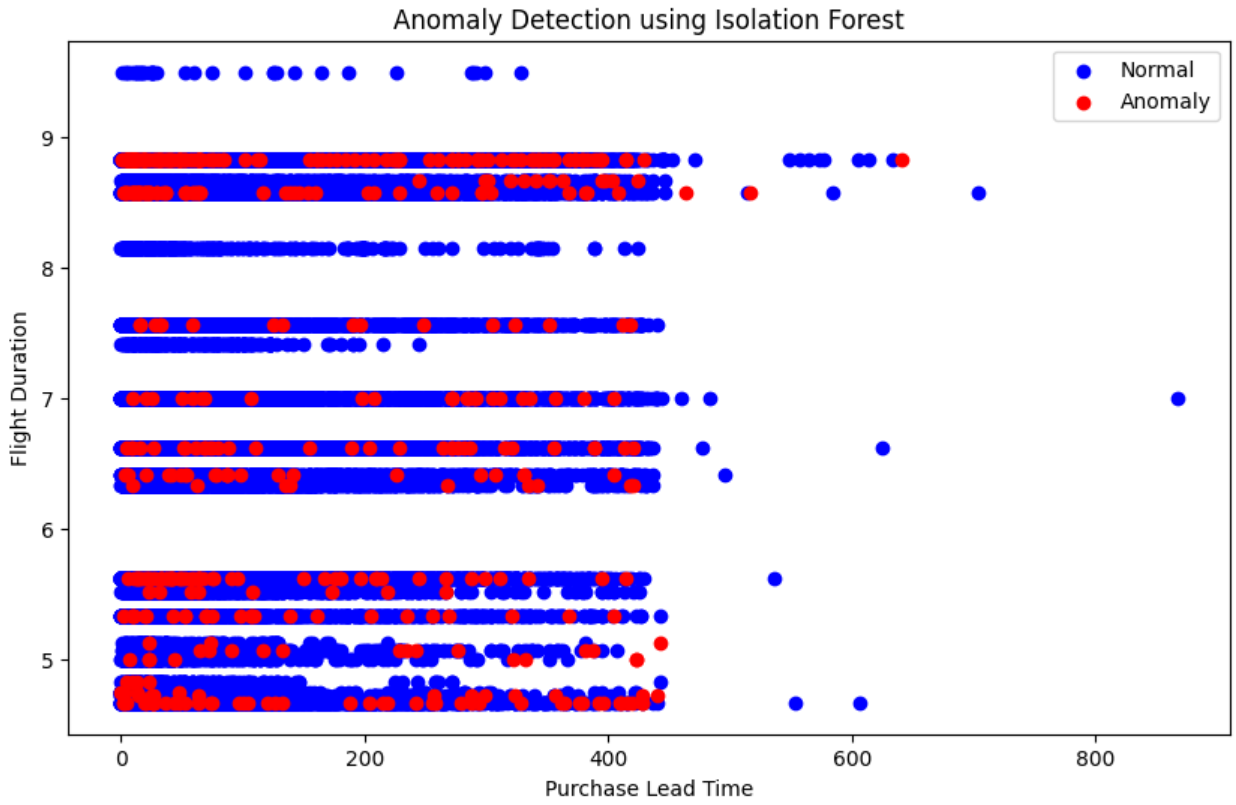


Anomaly Detection

```
features = ['num_passengers', 'purchase_lead', 'length_of_stay',
'flight_duration']
data = file[features]
```

Isolation Forest Model

```
from sklearn.ensemble import IsolationForest

ISF_MODEL = IsolationForest(contamination=0.01, random_state= 42)

ISF_MODEL.fit(data)

file['anomaly'] = ISF_MODEL.predict(data)

anomalies = file[file['anomaly'] == -1]
normal_data = file[file['anomaly'] == 1]

# Plot anomalies and normal data
plt.figure(figsize=(10, 6))
plt.scatter(normal_data['purchase_lead'],
normal_data['flight_duration'], c='blue', label='Normal')
plt.scatter(anomalies['purchase_lead'], anomalies['flight_duration'],
c='red', label='Anomaly')
plt.xlabel('Purchase Lead Time')
plt.ylabel('Flight Duration')
plt.title('Anomaly Detection using Isolation Forest')
plt.legend()
plt.show()
```




```
print("Anomalies found:")
print(anomalies)
```

```
Anomalies found:
       num_passengers sales_channel  trip_type  purchase_lead
length_of_stay  \
73                  1      Internet  RoundTrip            198
207
115                 1      Internet  RoundTrip             65
278
343                 6        Mobile  RoundTrip            349
21
426                 5      Internet  RoundTrip            343
43
479                 5      Internet  RoundTrip            301
34
...               ...           ...        ...            ...
...
49880               5      Internet  RoundTrip            409
6
49882               5      Internet  RoundTrip            376
6
49894               5      Internet  RoundTrip            364
6
49951               8      Internet  RoundTrip            133
```

```
6
49953               8      Internet  RoundTrip            328
6

       flight_hour flight_day   route booking_origin
wants_extra_baggage  \
73              10        Sun  AKLKIX          Japan
1
115              2        Thu  AKLKUL       Malaysia
0
343              5        Sun  AKLKUL       Malaysia
1
426              4        Thu  AKLKUL    New Zealand
1
479             10        Sun  AKLKUL    New Zealand
1
...            ...        ...     ...            ...
...
49880           11        Mon  PENTPE       Malaysia
1
49882            9        Sun  PENTPE       Malaysia
1
49894            6        Tue  PENTPE       Malaysia
1
49951            8        Wed  PENTPE       Malaysia
0
49953           17        Sat  PENTPE       Malaysia
1

       wants_preferred_seat  wants_in_flight_meals  flight_duration  \
73                        0                      1             7.00
115                       0                      0             8.83
343                       0                      0             8.83
426                       0                      0             8.83
479                       0                      0             8.83
...                     ...                    ...              ...
49880                     0                      1             4.67
49882                     0                      1             4.67
49894                     0                      0             4.67
49951                     0                      0             4.67
49953                     1                      1             4.67

       booking_complete  cluster  anomaly
73                    0        0       -1
115                   0        0       -1
343                   0        1       -1
426                   0        1       -1
479                   0        1       -1
...                 ...      ...      ...
49880                 1        1       -1
```

```
49882                    1         1        -1
49894                    1         1        -1
49951                    0         2        -1
49953                    1         1        -1

[500 rows x 16 columns]
```

BAR CHART DEPICTING THE TOP N ROUTES