In [1]:

```
import pandas as pd
from matplotlib.ticker import MaxNLocator
import matplotlib
from collections import Counter
import seaborn as sns
matplotlib.use('Qt5Agg')
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
tqdm.pandas()

## Data Cleaning & Exploration

# Load the spend and bookings CSV file into Pandas DataFrames
spend_data = pd.read_csv('sr_data_analyst_case_study_spend-new.csv', header=0)
bookings_data = pd.read_csv('sr_data_analyst_case_study_bookings-new.csv', header=0)

print(spend_data.head())
print(bookings_data.head())

# Clean up the last two empty columns of Spend CSV
unnamed_columns = spend_data.columns[spend_data.columns.str.startswith('Unnamed:')]
spend_data.drop(columns=unnamed_columns, inplace=True)
spend_data.reset_index(drop=True, inplace=True)

# Convert DATE columns in both dataframes from object to date
spend_data['DATE'] = pd.to_datetime(spend_data['DATE'])
bookings_data['DATE'] = pd.to_datetime(bookings_data['DATE'])

# Convert ELIGIBLE_IMPS, IMPRESSIONS and CLICKS from float to int and fill null values to 0
columns_to_convert = ['ELIGIBLE_IMPS', 'IMPRESSIONS', 'CLICKS']
spend_data[columns_to_convert]=spend_data[columns_to_convert].fillna(0)
spend_data[columns_to_convert]=spend_data[columns_to_convert].astype(int)

# Calculate median for COMPARISON_TO_LOWEST_PRICE ($)
median = spend_data['COMPARISON_TO_LOWEST_PRICE ($)'].median()
print('Median_COMPARISON_TO_LOWEST_PRICE:',median)

# Fill COMPARISON_TO_LOWEST_PRICE ($) missing values with its median
spend_data['COMPARISON_TO_LOWEST_PRICE ($)'].fillna(median, inplace=True)

# Replace missing values for spend to 0
spend_data['SPEND'].fillna(0, inplace=True)

# Add a numeric weekday column to bookings_data and spend_data
bookings_data['WEEKDAY'] = pd.to_datetime(bookings_data['DATE'], format='%A').dt.dayofweek
spend_data['WEEKDAY'] = pd.to_datetime(spend_data['DATE'], format='%A').dt.dayofweek

# Checking correlation between relevent columns
bookings_columns = ['LOS', 'DTA', 'BOOKINGS', 'GMV', 'WEEKDAY']
spend_columns = ['LOS', 'DTA', 'WEEKDAY', 'ELIGIBLE_IMPS', 'IMPRESSIONS', 'CLICKS', 'SPEND', 'COMPARISON_TO_LOWEST_PRICE ($)']
correlation_bookings = bookings_data[bookings_columns].corr()
correlation_spend = spend_data[spend_columns].corr()

print(correlation_bookings)
print(correlation_spend)

# # Print tables info
print(spend_data.info())
print(bookings_data.info())

# # Checking for null values
bookings_null_values = bookings_data.isnull().sum()
print(bookings_null_values)
spend_null_values = spend_data.isnull().sum()
print(spend_null_values)

# Output the distribution summary of some columns
print(spend_data['COMPARISON_TO_LOWEST_PRICE ($)'].describe())
print(spend_data['DATE'].describe())
print(bookings_data['DATE'].describe())
```

```
        DATE  HOTEL_ID  DTA  LOS  BOOKING_DOW  ELIGIBLE_IMPS  IMPRESSIONS  \
0 2020-03-12    112100   68    1          Thu              0            0
1 2020-03-13    551334    1    1          Fri              0            0
2 2020-03-14    145146    1    1          Sat              0            0
3 2020-03-11    456469    1    1          Wed              0            0
4 2020-03-12    233417  102    3          Thu              0            0

   CLICKS  SPEND  COMPARISON_TO_LOWEST_PRICE ($)  Unnamed: 10  Unnamed: 11
0       0    0.0                             NaN          NaN          NaN
1       0    0.0                             NaN          NaN          NaN
2       0    0.0                             NaN          NaN          NaN
3       0    0.0                             NaN          NaN          NaN
4       0    0.0                             NaN          NaN          NaN
        DATE  HOTEL_ID  DTA  LOS BOOKING_DOW  BOOKINGS          GMV
0 2020-03-15   109476    0    2         Sun         1    68.088832
1 2020-03-20   219156    0    1         Fri         1    64.857498
2 2020-03-15   105485    1    1         Sun         1   116.797901
3 2020-03-17   122473    0    1         Tue         1   118.256947
4 2020-03-15   201323    0    1         Sun         1    47.299482
Median_COMPARISON_TO_LOWEST_PRICE: 0.0435322864
                  LOS       DTA  BOOKINGS       GMV   WEEKDAY
LOS          1.000000  0.198000 -0.071284  0.610041 -0.028220
DTA          0.198000  1.000000 -0.040128  0.251534 -0.003159
BOOKINGS    -0.071284 -0.040128  1.000000  0.293561  0.043779
GMV          0.610041  0.251534  0.293561  1.000000  0.000011
WEEKDAY     -0.028220 -0.003159  0.043779  0.000011  1.000000
                                  LOS       DTA   WEEKDAY  ELIGIBLE_IMPS  \
LOS                          1.000000  0.226996  0.013665      -0.047928
DTA                          0.326996  1.000000  0.020160      -0.057701
WEEKDAY                      0.013665  0.020160  1.000000      -0.002644
ELIGIBLE_IMPS               -0.047928 -0.057701 -0.002644       1.000000
IMPRESSIONS                 -0.043699 -0.048756  0.008264       0.731385
CLICKS                      -0.010356 -0.043213  0.005775       0.269060
SPEND                       -0.003399 -0.046366  0.004881       0.161703
COMPARISON_TO_LOWEST_PRICE ($)  0.018709  0.031824  0.003457       0.004181

                            IMPRESSIONS    CLICKS     SPEND  \
LOS                       -0.042690 -0.010356 -0.003399
DTA                       -0.048756 -0.043213 -0.046366
WEEKDAY                    0.008264  0.005775  0.004881
ELIGIBLE_IMPS              0.731385  0.269060  0.161703
IMPRESSIONS                1.000000  0.346290  0.245742
CLICKS                     0.346290  1.000000  0.840962
SPEND                      0.245742  0.840962  1.000000
COMPARISON_TO_LOWEST_PRICE ($)  -0.011281 -0.088222 -0.086862

                            COMPARISON_TO_LOWEST_PRICE ($)
LOS                                               0.018709
DTA                                               0.031824
WEEKDAY                                           0.003457
ELIGIBLE_IMPS                                     0.004181
IMPRESSIONS                                      -0.011281
CLICKS                                           -0.088222
SPEND                                            -0.086862
COMPARISON_TO_LOWEST_PRICE ($)                    1.000000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 11 columns):
 #   Column                         Non-Null Count    Dtype
---  ------                         --------------    -----
 0   DATE                           1048575 non-null  datetime64[ns]
 1   HOTEL_ID                       1048575 non-null  int64
 2   DTA                            1048575 non-null  int64
 3   LOS                            1048575 non-null  int64
 4   BOOKING_DOW                    1048575 non-null  object
 5   ELIGIBLE_IMPS                  1048575 non-null  int64
 6   IMPRESSIONS                    1048575 non-null  int64
 7   CLICKS                         1048575 non-null  int64
 8   SPEND                          1048575 non-null  float64
 9   COMPARISON_TO_LOWEST_PRICE ($)  1048575 non-null  float64
 10  WEEKDAY                        1048575 non-null  int32
dtypes: datetime64[ns](1), float64(2), int32(1), int64(6), object(1)
memory usage: 84.0+ MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54912 entries, 0 to 54911
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   DATE         54912 non-null  datetime64[ns]
 1   HOTEL_ID     54912 non-null  int64
 2   DTA          54912 non-null  int64
 3   LOS          54912 non-null  int64
 4   BOOKING_DOW  54912 non-null  object
 5   BOOKINGS     54912 non-null  int64
 6   GMV          54912 non-null  float64
 7   WEEKDAY      54912 non-null  int32
dtypes: datetime64[ns](1), float64(1), int32(1), int64(4), object(1)
memory usage: 3.1+ MB
None
DATE                             0
HOTEL_ID                         0
DTA                              0
LOS                              0
BOOKING_DOW                      0
BOOKINGS                         0
GMV                              0
WEEKDAY                          0
dtype: int64
DATE                             0
HOTEL_ID                         0
DTA                              0
LOS                              0
BOOKING_DOW                      0
ELIGIBLE_IMPS                    0
IMPRESSIONS                      0
CLICKS                           0
SPEND                            0
COMPARISON_TO_LOWEST_PRICE ($)   0
WEEKDAY                          0
dtype: int64
count    1.048575e+06
mean     4.584382e-02
std      1.595523e-01
min     -7.101963e+01
25%      4.353286e-02
50%      4.353286e-02
75%      4.353286e-02
max      9.955575e-01
Name: COMPARISON_TO_LOWEST_PRICE ($), dtype: float64
count                  1048575
mean     2020-03-10 16:41:31.701595392
min                2020-03-01 00:00:00
25%                2020-03-05 00:00:00
50%                2020-03-09 00:00:00
75%                2020-03-15 00:00:00
max                2020-03-30 00:00:00
Name: DATE, dtype: object
count                    54912
mean     2020-03-13 16:39:09.125874176
min                2020-03-01 00:00:00
25%                2020-03-07 00:00:00
50%                2020-03-13 00:00:00
75%                2020-03-20 00:00:00
max                2020-03-30 00:00:00
Name: DATE, dtype: object
```

In [2]:

```
# Define an itinerary by the combination of 'DTA', 'LOS', 'BOOKING_DOW'
bookings_data['ITINERARY'] = bookings_data['DTA'].astype(str) + '_' + bookings_data['LOS'].astype(str) + '_' + bookings_data['BOOKING_DOW']

# Find the top itineraries based on the frequency of occurrence
top_itineraries = bookings_data['ITINERARY'].value_counts().nlargest(10).index

# Extract the top itineraries data
top_itineraries_data = bookings_data[bookings_data['ITINERARY'].isin(top_itineraries)]

# Calculate the average price per booking for top itineraries
top_itineraries_avg_price = top_itineraries_data['GMV'].mean()

# Calculate the average price per booking specifically for Friday within the top itineraries
top_friday_itineraries_data = top_itineraries_data[top_itineraries_data['BOOKING_DOW'] == 'Fri']
friday_avg_price = top_friday_itineraries_data['GMV'].mean()

# Calculate the most common day of week within the top itineraries
top_itineraries_dow = top_itineraries_data['BOOKING_DOW'].mode()[0]

# Output the recalculated results
print('Recalculated average price per booking for top itineraries:', top_itineraries_avg_price)
print('Recalculated average price per booking for top itineraries on Friday:', friday_avg_price)
print('The most common day of week within the top itineraries is:', top_itineraries_dow)

# Define epsilon to avoid division by zero errors
epsilon = 1e-8

# Calculate the overall impression to eligible impression rate
overall_imp_eligible_rate = spend_data['IMPRESSIONS'].sum() / (spend_data['ELIGIBLE_IMPS'].sum() + epsilon)

# Calculate the overall conversion rate (clicks to impression rate)
overall_click_rate = spend_data['CLICKS'].sum() / (spend_data['IMPRESSIONS'].sum() + epsilon)

# Display the overall rates
print('Overall Impression/Eligible Impression Rate:', overall_imp_eligible_rate)
print('Overall Click Rate:', overall_click_rate)

# Get the count of bookings by Day of Week for the top itineraries, sorted by count
dow_counts_sorted = top_itineraries_data['BOOKING_DOW'].value_counts().sort_values(ascending=False)

# Plotting
plt.figure(figsize=(10, 6))
bar_plot = dow_counts_sorted.plot(kind='bar', color='hotpink')
plt.title('Distribution of Bookings by Day of Week for Top Itineraries (Sorted)')
plt.xlabel('Day of Week', color='black')
plt.ylabel('Number of Bookings', color='black')
plt.xticks(rotation=45, color='black')
plt.yticks(color='black')

# Adding the labels on top of the bars
for p in bar_plot.patches:
    bar_plot.annotate(str(p.get_height()), (p.get_x() * 1.005, p.get_height() * 1.005), color='black')

plt.tight_layout()
#plt.show()

# Calculate the average GMV by day of week for top itineraries
top_itineraries_avg_gmv_by_dow = top_itineraries_data.groupby('BOOKING_DOW')['GMV'].mean()

# Sort the average GMV from high to low
top_itineraries_avg_gmv_by_dow_sorted = top_itineraries_avg_gmv_by_dow.sort_values(ascending=False)


# Filter rows where eligible impressions are greater than actual impressions
spend_filtered = spend_data[spend_data['ELIGIBLE_IMPS'] > spend_data['IMPRESSIONS']]

# # Display the head of the filtered dataframe to show a preview of the data
# print(spend_filtered.head())

plt.figure(figsize=(10, 6))
chart = sns.countplot(x='BOOKING_DOW', data=spend_filtered, palette='hotpink', order = spend_filtered['BOOKING_DOW'].value_counts().index)
chart.set_title('Frequency by Day of Week for Eligible Impressions Greater than Actual Impressions')
chart.set_xlabel('Day of Week')
chart.set_ylabel('Frequency')

# Add value labels to each bar
for p in chart.patches:
    chart.annotate(format(p.get_height(), '.0f'),
                   (p.get_x() + p.get_width() / 2., p.get_height()),
                   ha = 'center', va = 'center',
                   xytext = (0, 9),
                   textcoords = 'offset points')

# Improve the readability of the labels
chart.set_xticklabels(chart.get_xticklabels(), rotation=45)

# Ensure that the y-axis only shows integer ticks
chart.yaxis.set_major_locator(MaxNLocator(integer=True))

plt.tight_layout()
#plt.show()

# Plotting
plt.figure(figsize=(12, 8))
bar_plot = top_itineraries_avg_gmv_by_dow_sorted.plot(kind='bar', color='hotpink')
plt.title('Average GMV by Day of Week for Top Itineraries (High to Low)')
plt.xlabel('Day of Week')
plt.ylabel('Average GMV in USD')
plt.xticks(rotation=45)

# Adding the labels on top of the bars with $ sign
for p in bar_plot.patches:
    bar_plot.annotate(f'$' + f'{p.get_height():.2f}', (p.get_x() * 1.005, p.get_height() * 1.005))

plt.tight_layout()
#plt.show()
```

```
Recalculated average price per booking for top itineraries: 86.00747365676358
Recalculated average price per booking for top itineraries on Friday: 100.46494548457173
The most common day of week within the top itineraries is: Fri
Overall Impression/Eligible Impression Rate: 0.4838193143545284
Overall Click Rate: 0.031871626243495997
```

In [ ]:

```
# Analyze the relationship between DTA and GMV
# Group the data by DTA and calculate the average GMV for each group
avg_gmv_by_dta = bookings_data.groupby('DTA')['GMV'].mean().reset_index()

# Plot the relationship between DTA and average GMV
plt.figure(figsize=(12, 6))
sns.lineplot(x='DTA', y='GMV', data=avg_gmv_by_dta)
plt.title('Average GMV by Days to Arrival (DTA)')
plt.xlabel('Days to Arrival (DTA)')
plt.ylabel('Average GMV')
plt.xlim(0, 30) # Limiting to 30 days for better visibility
plt.show()
```