Importing Libraries

In [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:
```
lead = pd.read_csv('Data_Science_Internship - Dump.csv', sep=',')
```

In [5]:
```
lead.head(20)
```

Out[5]:

| | Unnamed: 0 | Agent_id | status | lost_re |
|---|---|---|---|---|
| **0** | 0 | 1deba9e96f404694373de9749ddd1ca8aa7bb823145a6f... | LOST | Not respor |
| **1** | 1 | 299ae77a4ef350ae0dd37d6bba1c002d03444fb1edb236... | LOST | Low bu |
| **2** | 2 | c213697430c006013012dd2aca82dd9732aa0a1a6bca13... | LOST | Not respor |
| **3** | 3 | eac9815a500f908736d303e23aa227f0957177b0e6756b... | LOST | Low bu |
| **4** | 4 | 1deba9e96f404694373de9749ddd1ca8aa7bb823145a6f... | LOST | Junk |
| **5** | 5 | 2306878a9ad9b57686cd623dd285aaa9b25afdf627f651... | LOST | Wants pr accommod |
| **6** | 6 | 2306878a9ad9b57686cd623dd285aaa9b25afdf627f651... | LOST | Short |
| **7** | 7 | 44864c96fa1c36602f0d045b268981b6cab638a60fc207... | LOST | Wants pr accommod |
| **8** | 8 | ab6bb4584e9946b135dca2e39d12abba3ea82d5ea927d0... | LOST | Low bu |
| **9** | 9 | 131127203c89e8219dbdfe2f597538759310f40918b222... | LOST | Booked mar |
| **10** | 10 | 2fca346db656187102ce806ac732e06a62df0dbb2829e5... | LOST | Low availa |
| **11** | 11 | d4192f06768ab0f257c7f5e17ad021b075b995d4a18675... | LOST | Not respor |
| **12** | 12 | 44864c96fa1c36602f0d045b268981b6cab638a60fc207... | LOST | Wants pr accommod |

| | | | | |
|---|---|---|---|---|
| **13** | 13 | 50750ee66f27656c2b34d43078a064c3b9b8807938b6a3... | LOST | Not respor |
| **14** | 14 | a9f80b4eaba3fd134bafafe7506e08940201964615f7ee... | LOST | Low bu |
| **15** | 15 | a9f80b4eaba3fd134bafafe7506e08940201964615f7ee... | LOST | Not respor |
| **16** | 16 | 2fca346db656187102ce806ac732e06a62df0dbb2829e5... | LOST | Low availa |
| **17** | 17 | f1ece3b02f1e5989bb0918e468fbc3f3e60d74ed90809d... | LOST | Wants pr accommod |
| **18** | 18 | f1ece3b02f1e5989bb0918e468fbc3f3e60d74ed90809d... | LOST | Junk |
| **19** | 19 | 2fca346db656187102ce806ac732e06a62df0dbb2829e5... | LOST | Low availa |

EDA

In [6]:
```
lead.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46608 entries, 0 to 46607
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Unnamed: 0      46608 non-null  int64 
 1   Agent_id        46608 non-null  object
 2   status          46608 non-null  object
 3   lost_reason     43244 non-null  object
 4   budget          42908 non-null  object
 5   lease           44267 non-null  object
 6   movein          32970 non-null  object
 7   source          46608 non-null  object
 8   source_city     46608 non-null  object
 9   source_country  46608 non-null  object
 10  utm_source      46608 non-null  object
 11  utm_medium      46608 non-null  object
 12  des_city        46608 non-null  object
 13  des_country     46608 non-null  object
 14  room_type       23061 non-null  object
 15  lead_id         46608 non-null  object
dtypes: int64(1), object(15)
memory usage: 5.7+ MB
```

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

Out[7]:

| | Unnamed: 0 |
|---|---|
| **count** | 46608.00000 |
| **mean** | 23303.50000 |
| **std** | 13454.71501 |
| **min** | 0.00000 |
| **25%** | 11651.75000 |
| **50%** | 23303.50000 |
| **75%** | 34955.25000 |
| **max** | 46607.00000 |

In [8]:
```
l = lead.copy()
```

# DATA CLEANING

In [9]:
```
#Dropped Unnamed Column because we already have index
l.drop(['Unnamed: 0'], axis=1, inplace=True)
```

In [10]:
```
l = pd.DataFrame(l)
l3=l.replace(to_replace="9b2d5b4678781e53038e91ea5324530a03f27dc1d0e5f6c9
           value= np.nan)
```

In [11]:
```
l3.isnull().sum()
```

Out[11]:
```
Agent_id              0
status                0
lost_reason        3364
budget             3700
lease              2341
movein            13638
source             5977
source_city        8851
source_country     8641
utm_source           61
utm_medium         3187
des_city           2537
des_country        2537
room_type         23547
lead_id               0
dtype: int64
```

In [12]:
```
print("Number of Categories in: ")
for ColName in l[['lost_reason','budget','lease','movein','source','sourc
                  'utm_medium','des_city','des_country','room_type','lead
    print("{} = {}".format(ColName,       len(l[ColName].unique())))
```

```
Number of Categories in:
lost_reason = 31
budget = 1858
lease = 312
movein = 478
source = 683
source_city = 4336
source_country = 186
utm_source = 35
utm_medium = 64
des_city = 220
des_country = 15
room_type = 6
lead_id = 30574
```

Replacing NAN values with the most frequent occurred category in variable/column.

We replaced every column that had missing values except the columns 'movein' and 'room_type' because replacing it with mode value will bias prediction .

In [13]:
```
# Function to replace NAN values with mode value
def impute_nan_most_frequent_category(l3,ColName):

     most_frequent_category=l3[ColName].mode()[0]

# replace nan values with most occured category
     l3[ColName + "_Imputed"] = l3[ColName]
     l3[ColName + "_Imputed"].fillna(most_frequent_category,inplace=True)

for Columns in ['lost_reason','budget','lease','source','source_city','so
                  'utm_medium','des_city','des_country']:
    impute_nan_most_frequent_category(l3,Columns)


l3[['lost_reason','lost_reason_Imputed','budget','budget_Imputed','lease'
   ,'source_country','source_country_Imputed','utm_source','utm_source_Im
   ,'des_country','des_country_Imputed']].head(10)

l3 = l3.drop(['lost_reason','budget','lease','source','source_city','sour
                  'utm_medium','des_city','des_country'], axis = 1)
```

In [14]:
```
l3.rename(columns = {'status':'OUTPUT'}, inplace = True)
l3
```

Out[14]:

| | Agent_id | OUTPUT | movein | room_t |
|---|---|---|---|---|
| **0** | 1deba9e96f404694373de9749ddd1ca8aa7bb823145a6f... | LOST | NaN | |
| **1** | 299ae77a4ef350ae0dd37d6bba1c002d03444fb1edb236... | LOST | NaN | |
| **2** | c213697430c006013012dd2aca82dd9732aa0a1a6bca13... | LOST | 31/08/22 | Ens |
| **3** | eac9815a500f908736d303e23aa227f0957177b0e6756b... | LOST | NaN | |
| **4** | 1deba9e96f404694373de9749ddd1ca8aa7bb823145a6f... | LOST | NaN | |
| **...** | ... | ... | ... | |
| **46603** | 2306878a9ad9b57686cd623dd285aaa9b25afdf627f651... | LOST | 01/09/22 | St |
| **46604** | 327ec29056cc47c24bf922f7dc0f78261dad5c726d7353... | LOST | 29/09/22 | St |
| **46605** | 1134c0a7d44fdae1afd7f1f64e2789496784095ca0a050... | LOST | 20/09/22 | St |
| **46606** | 8b8b029f1142f5cbc825aa6cbee01406c915c6b055db79... | LOST | 30/08/22 | |
| **46607** | 1ea65ea38f2f574b3875ba895e4ff76b284b7725041612... | LOST | 01/09/22 | St |

46608 rows × 15 columns

Replacing NAN categories with most occurred values, and adding a new feature to introduce some weight/importance to non-imputed and imputed observations.

In [15]:
```
# Function to impute most occured category and add importance vairable
def impute_nan_add_vairable(l3,ColName):
    #1. add new column and replace if category is null then 1 else 0
    l3[ColName+"_Imputed"] =   np.where(l3[ColName].isnull(),1,0)

    #Take most occured category in that vairable (.mode())

    Mode_Category = l3[ColName].mode()[0]

    #Replace NAN values with most occured category in actual vairable

    l3[ColName].fillna(Mode_Category,inplace=True)
# Call function to impute NAN values and add new importance feature
for Columns in ['movein','room_type']:
    impute_nan_add_vairable(l3,Columns)

# Display top 10 row to see the result of imputation
l3[['movein','movein_Imputed','room_type','room_type_Imputed']].head(10)
```

Out[15]:

| | movein | movein_Imputed | room_type | room_type_Imputed |
|---|---|---|---|---|
| **0** | 10/09/22 | 1 | Ensuite | 1 |
| **1** | 10/09/22 | 1 | Ensuite | 1 |
| **2** | 31/08/22 | 0 | Ensuite | 0 |
| **3** | 10/09/22 | 1 | Ensuite | 1 |
| **4** | 10/09/22 | 1 | Ensuite | 1 |
| **5** | 10/09/22 | 1 | Ensuite | 1 |
| **6** | 10/09/22 | 1 | Ensuite | 1 |
| **7** | 08/09/22 | 0 | Entire Place | 0 |
| **8** | 10/09/22 | 1 | Ensuite | 1 |
| **9** | 10/09/22 | 1 | Ensuite | 1 |

In [16]:
```
l3 = l3.drop(['movein','room_type'], axis = 1)
```

In [17]:
```
l3
```

Out[17]:

| | Agent_id | OUTPUT | |
|---|---|---|---|
| 0 | 1deba9e96f404694373de9749ddd1ca8aa7bb823145a6f... | LOST | cd5dc0d9393f3 |
| 1 | 299ae77a4ef350ae0dd37d6bba1c002d03444fb1edb236... | LOST | b94693673a5f717 |
| 2 | c213697430c006013012dd2aca82dd9732aa0a1a6bca13... | LOST | 96ea4e2bf04496c |
| 3 | eac9815a500f908736d303e23aa227f0957177b0e6756b... | LOST | 1d2b34d8add02a1 |
| 4 | 1deba9e96f404694373de9749ddd1ca8aa7bb823145a6f... | LOST | fc10fffd29cfbe |
| ... | ... | ... | |
| 46603 | 2306878a9ad9b57686cd623dd285aaa9b25afdf627f651... | LOST | 1aaa4a4a9092e4 |
| 46604 | 327ec29056cc47c24bf922f7dc0f78261dad5c726d7353... | LOST | 1f90dbad4873cl |
| 46605 | 1134c0a7d44fdae1afd7f1f64e2789496784095ca0a050... | LOST | d9e0f455b68a6 |
| 46606 | 8b8b029f1142f5cbc825aa6cbee01406c915c6b055db79... | LOST | 1f90dbad4873cl |
| 46607 | 1ea65ea38f2f574b3875ba895e4ff76b284b7725041612... | LOST | 7520a8abba2b44 |

46608 rows × 15 columns

In [18]:

```
l3.isnull().sum()
```

Out[18]:
```
Agent_id                  0
OUTPUT                    0
lead_id                   0
lost_reason_Imputed       0
budget_Imputed            0
lease_Imputed             0
source_Imputed            0
source_city_Imputed       0
source_country_Imputed    0
utm_source_Imputed        0
utm_medium_Imputed        0
des_city_Imputed          0
des_country_Imputed       0
movein_Imputed            0
room_type_Imputed         0
dtype: int64
```

No Null Values

In [19]:
```
l3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46608 entries, 0 to 46607
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Agent_id                46608 non-null  object
 1   OUTPUT                  46608 non-null  object
 2   lead_id                 46608 non-null  object
 3   lost_reason_Imputed     46608 non-null  object
 4   budget_Imputed          46608 non-null  object
 5   lease_Imputed           46608 non-null  object
 6   source_Imputed          46608 non-null  object
 7   source_city_Imputed     46608 non-null  object
 8   source_country_Imputed  46608 non-null  object
 9   utm_source_Imputed      46608 non-null  object
 10  utm_medium_Imputed      46608 non-null  object
 11  des_city_Imputed        46608 non-null  object
 12  des_country_Imputed     46608 non-null  object
 13  movein_Imputed          46608 non-null  int64 
 14  room_type_Imputed       46608 non-null  int64 
dtypes: int64(2), object(13)
memory usage: 5.3+ MB
```

In [20]:
```
#Dropping rows with values for OUTPUT is not WON or LOST
l3['movein_Imputed '] = l3.movein_Imputed .astype('object')
l3['room_type_Imputed '] = l3.room_type_Imputed .astype('object')
l3 = l3.drop(['movein_Imputed','room_type_Imputed'], axis = 1)


status_drop = l3[ (l3['OUTPUT'] != 'WON') & (l3['OUTPUT'] != 'LOST') ].in
l3.drop(status_drop, inplace=True)

l3.dtypes
```

Out[20]:
```
Agent_id                  object
OUTPUT                    object
lead_id                   object
lost_reason_Imputed       object
budget_Imputed            object
lease_Imputed             object
source_Imputed            object
source_city_Imputed       object
source_country_Imputed    object
utm_source_Imputed        object
utm_medium_Imputed        object
des_city_Imputed          object
des_country_Imputed       object
movein_Imputed            object
room_type_Imputed         object
dtype: object
```

In [21]:
```
l3.columns
```

Out[21]:
```
Index(['Agent_id', 'OUTPUT', 'lead_id', 'lost_reason_Imputed',
       'budget_Imputed', 'lease_Imputed', 'source_Imputed',
       'source_city_Imputed', 'source_country_Imputed', 'utm_source_Input
ed',
       'utm_medium_Imputed', 'des_city_Imputed', 'des_country_Imputed',
       'movein_Imputed ', 'room_type_Imputed '],
      dtype='object')
```

Using LabelEncoder because One-Hot Encoding will use too much space and each feature has too many categories

In [22]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
categ = ['Agent_id', 'OUTPUT', 'lead_id', 'lost_reason_Imputed',
       'budget_Imputed', 'lease_Imputed', 'source_Imputed',
       'source_city_Imputed', 'source_country_Imputed', 'utm_source_Input
       'utm_medium_Imputed', 'des_city_Imputed', 'des_country_Imputed',
       'movein_Imputed ', 'room_type_Imputed ']



l3[categ] = l3[categ].apply(label_encoder.fit_transform)
```

In [ ]:

In [ ]:

In [23]:
```
l3['OUTPUT'] = l3['OUTPUT'].multiply(100)
```

In [24]:
```
l3.head(100)
```

Out[24]:

| | Agent_id | OUTPUT | lead_id | lost_reason_Imputed | budget_Imputed | lease_Imputed |
|---|---|---|---|---|---|---|
| **0** | 12 | 0 | 24421 | 21 | 1834 | 266 |
| **1** | 20 | 0 | 22037 | 16 | 1834 | 266 |
| **2** | 87 | 0 | 17924 | 21 | 1752 | 266 |
| **3** | 110 | 0 | 3582 | 16 | 10 | 2 |
| **4** | 12 | 0 | 30016 | 8 | 1834 | 266 |
| **...** | ... | ... | ... | ... | ... | ... |
| **95** | 18 | 0 | 1615 | 21 | 1752 | 266 |
| **96** | 86 | 0 | 23864 | 21 | 1793 | 266 |
| **97** | 23 | 0 | 10081 | 21 | 1834 | 263 |
| **98** | 86 | 100 | 21687 | 15 | 806 | 197 |
| **99** | 98 | 0 | 1615 | 16 | 1834 | 278 |

100 rows × 15 columns

In [ ]:

```
```

In [25]:

```
l3["OUTPUT"].value_counts()

## Data Is Highly Imbalanced
```

Out[25]:

```
0      43244
100     3073
Name: OUTPUT, dtype: int64
```

In [60]:

```
# Scaling The Data Between 0-100, just to make predictions better.

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 100))


scaler.fit(l3)
```

Out[60]:

```
MinMaxScaler(feature_range=(0, 100))
```

In [27]:

```
scaled_data = scaler.transform(l3)
l4 = pd.DataFrame(scaled_data, columns = categ)
```

In [28]:

```
##PCA - Principal Component Analysis-
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
pca.fit(scaled_data)
x_pca = pca.transform(scaled_data)

x_pca.shape
```

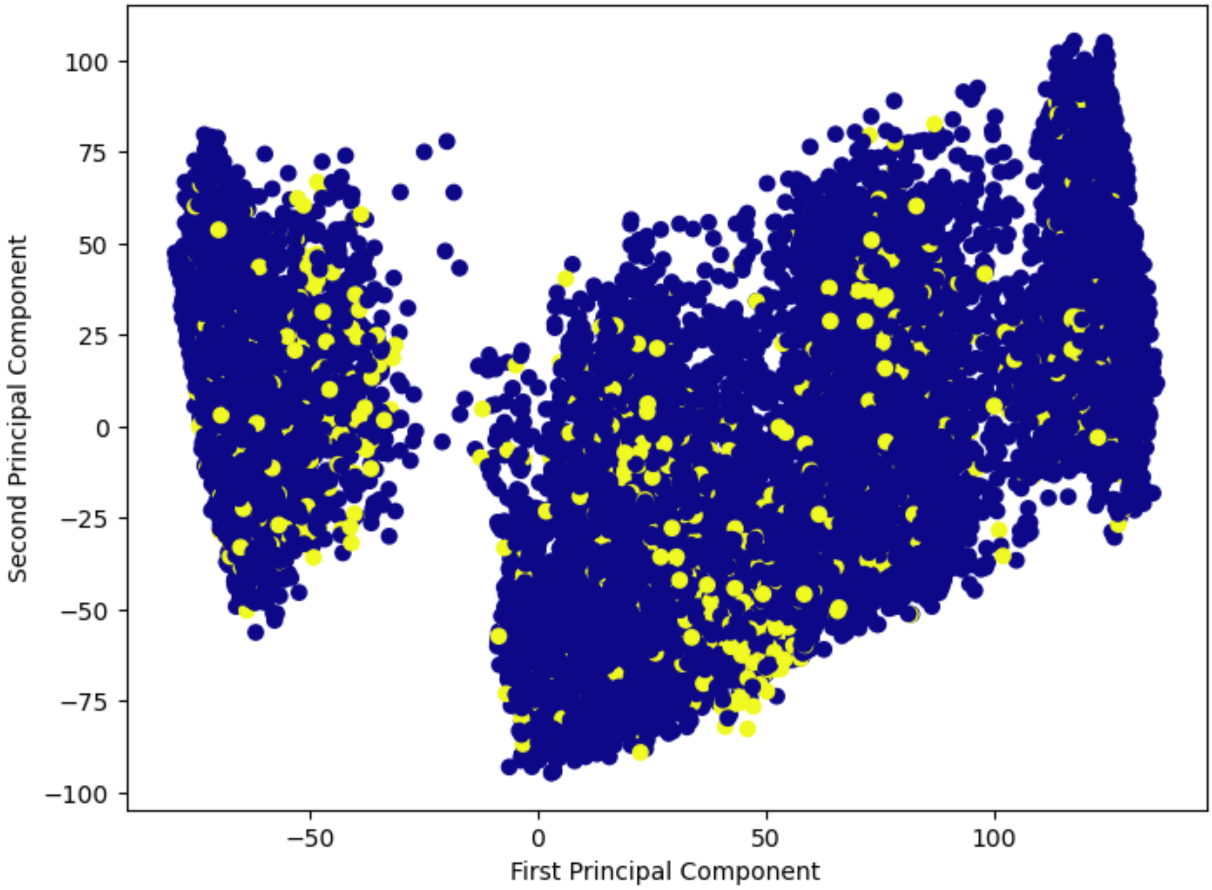
Out[28]: (46317, 2)

In [29]:
```
plt.figure(figsize =(8, 6))

plt.scatter(x_pca[:, 0], x_pca[:, 1], c = l4['OUTPUT'], cmap ='plasma')

# labeling x and y axes
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
```

Out[29]: Text(0, 0.5, 'Second Principal Component')




In [30]:
```
pca.components_
```

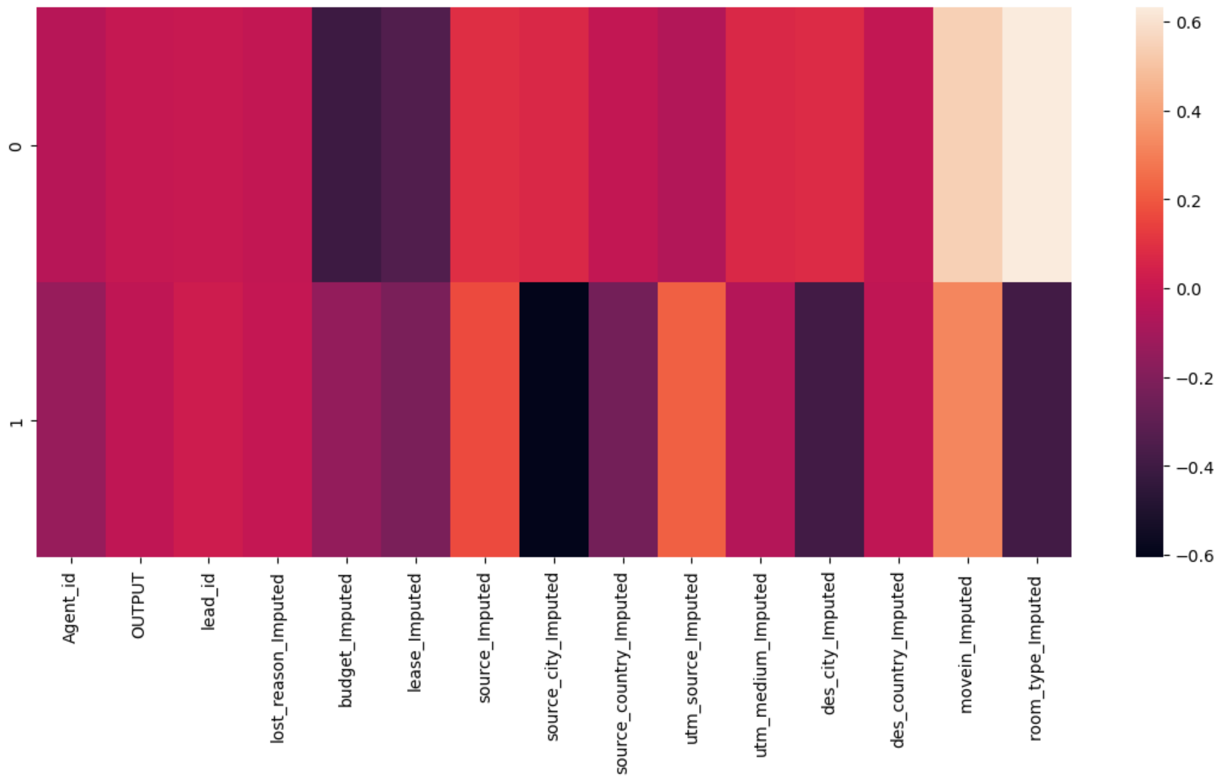
Out[30]:
```
array([[-0.04619647, -0.00296846,  0.00241814, -0.01500878, -0.40406981,
        -0.34162603,  0.08592064,  0.0666444 , -0.01173579, -0.06260317,
         0.06943671,  0.07578934, -0.01431473,  0.54055576,  0.63150104],
       [-0.14151827, -0.0241828 ,  0.01833759, -0.00944847, -0.1504888 ,
        -0.2197464 ,  0.16731947, -0.60562223, -0.24060112,  0.21728732,
        -0.05484478, -0.39009715, -0.02300242,  0.31883631, -0.38830486]]
)
```

In [31]:
```
df_comp = pd.DataFrame(pca.components_, columns = categ)

plt.figure(figsize =(14, 6))

# plotting heatmap
sns.heatmap(df_comp)
```

Out[31]: <AxesSubplot:>

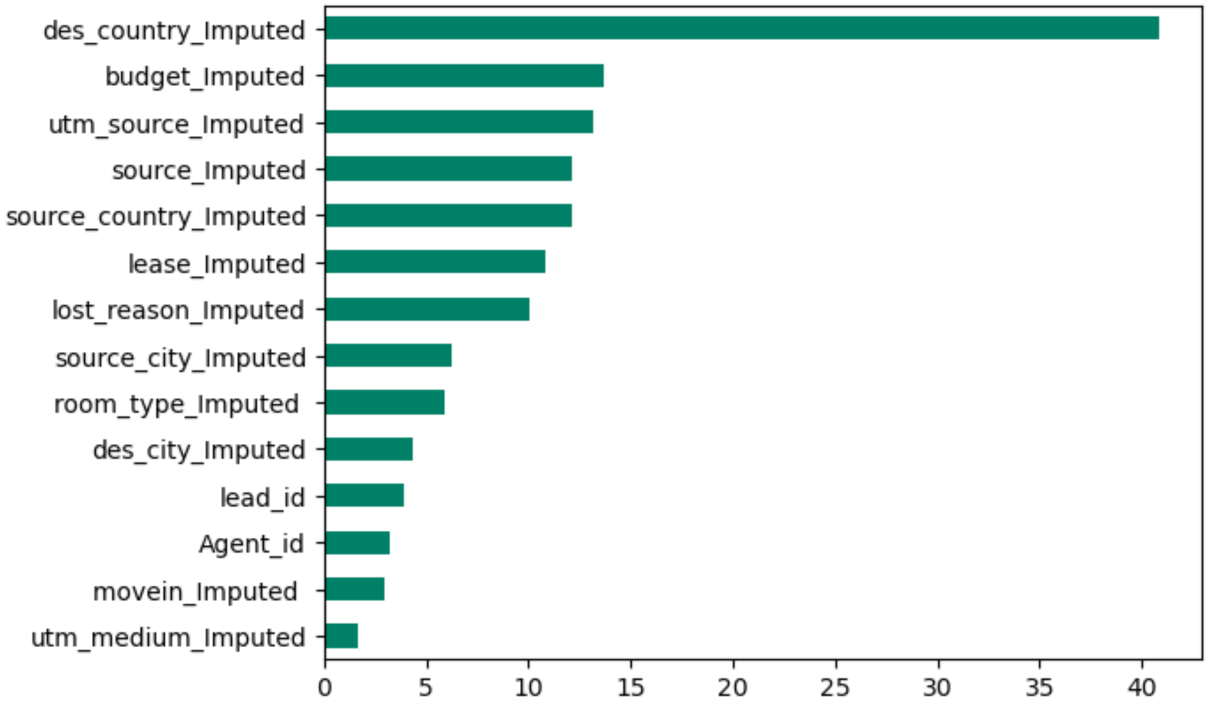



# FEATURE SELECTION

In [32]:
```
X=l4.drop("OUTPUT",axis=1)
Y=l4["OUTPUT"]

from statsmodels.stats.outliers_influence import variance_inflation_facto
_ = l4[['Agent_id', 'lead_id', 'lost_reason_Imputed',
        'budget_Imputed', 'lease_Imputed', 'source_Imputed',
        'source_city_Imputed', 'source_country_Imputed', 'utm_source_Imput
        'utm_medium_Imputed', 'des_city_Imputed', 'des_country_Imputed',
        'movein_Imputed ', 'room_type_Imputed ']]
vif_lst = []
for i in range(_.shape[1]):
    vif = variance_inflation_factor(_.to_numpy(),i)
    vif_lst.append(vif)
s1 = pd.Series(vif_lst, index = _.columns)
s1.sort_values().plot(kind = "barh", cmap = "summer")
```

Out[32]: <AxesSubplot:>




All Features are independent from each other

In [33]:
```
from skfeature.function.similarity_based import fisher_score
#fisher_rank = fisher_score.fisher_score(_.to_numpy(),Y)
s = pd.Series(fisher_rank, index = _.columns)
s.sort_values().plot(kind = "barh")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/bt/rtt5pc916cq7xtghg828tf800000gp/T/ipykernel_8095/3609299255.py in <module>
      1 from skfeature.function.similarity_based import fisher_score
      2 #fisher_rank = fisher_score.fisher_score(_.to_numpy(),Y)
----> 3 s = pd.Series(fisher_rank, index = _.columns)
      4 s.sort_values().plot(kind = "barh")

NameError: name 'fisher_rank' is not defined
```

In [34]:
```
from sklearn.feature_selection import mutual_info_classif, chi2, f_classi
var_th = VarianceThreshold(threshold = 0.0)
var_th.fit_transform(_)
s = pd.Series(var_th.get_support(),index = _.columns)
s
```

Out[34]:
```
Agent_id                  True
lead_id                   True
lost_reason_Imputed       True
budget_Imputed            True
lease_Imputed             True
source_Imputed            True
source_city_Imputed       True
source_country_Imputed    True
utm_source_Imputed        True
utm_medium_Imputed        True
des_city_Imputed          True
des_country_Imputed       True
movein_Imputed            True
room_type_Imputed         True
dtype: bool
```

ALL ARE GOOD PREDICTORS

In [35]:
```
_temp = l4[['Agent_id', 'lead_id', 'lost_reason_Imputed',
       'budget_Imputed', 'lease_Imputed', 'source_Imputed',
       'source_city_Imputed', 'source_country_Imputed', 'utm_source_Imput
       'utm_medium_Imputed', 'des_city_Imputed', 'des_country_Imputed',
       'movein_Imputed ', 'room_type_Imputed ']]
chi, p_val = chi2(_temp,l3["OUTPUT"])
s = pd.DataFrame({"Chi2":np.around(chi,2), "P_val": np.around(p_val,2)},
s
```

Out[35]:

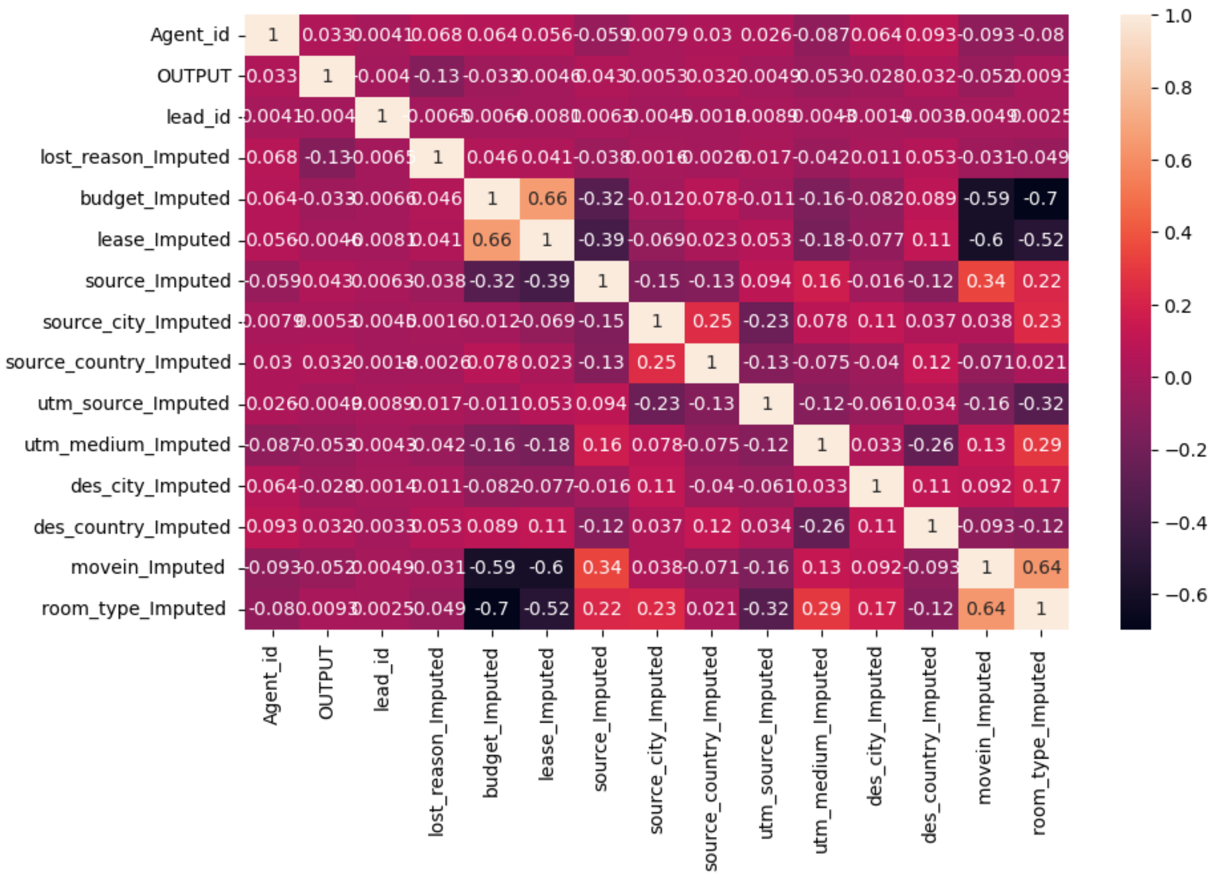
| | Chi2 | P_val |
|---|---|---|
| **Agent_id** | 1048.57 | 0.00 |
| **lead_id** | 12.58 | 0.00 |
| **lost_reason_Imputed** | 4897.31 | 0.00 |
| **budget_Imputed** | 867.07 | 0.00 |
| **lease_Imputed** | 15.81 | 0.00 |
| **source_Imputed** | 480.31 | 0.00 |
| **source_city_Imputed** | 19.02 | 0.00 |
| **source_country_Imputed** | 359.52 | 0.00 |
| **utm_source_Imputed** | 6.52 | 0.01 |
| **utm_medium_Imputed** | 3641.90 | 0.00 |
| **des_city_Imputed** | 659.58 | 0.00 |
| **des_country_Imputed** | 52.42 | 0.00 |
| **movein_Imputed** | 8774.50 | 0.00 |
| **room_type_Imputed** | 196.25 | 0.00 |

room_type and utm_source isnt that important for our model

In [36]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
# Correlation matrix
cor= l4.corr()
# PLoan
plt.figure(figsize = (10,6))
sns.heatmap(cor, annot = True)
```

Out[36]: <AxesSubplot:>



We observe colums movein and room_type are co-related

In [37]:

```
_temp = l4[['Agent_id', 'lead_id', 'lost_reason_Imputed',
        'budget_Imputed', 'lease_Imputed', 'source_Imputed',
        'source_city_Imputed', 'source_country_Imputed', 'utm_source_Imput
        'utm_medium_Imputed', 'des_city_Imputed', 'des_country_Imputed',
        'movein_Imputed ', 'room_type_Imputed ']]
f_val,p_val = f_classif(_temp,l3["OUTPUT"])
pd.DataFrame({"F_Val": np.around(f_val,2), "P_val": np.around(p_val,2)},i
```

Out[37]:

| | F_Val | P_val |
|---|---|---|
| **Agent_id** | 50.61 | 0.00 |
| **lead_id** | 0.75 | 0.39 |
| **lost_reason_Imputed** | 772.38 | 0.00 |
| **budget_Imputed** | 49.87 | 0.00 |
| **lease_Imputed** | 0.98 | 0.32 |
| **source_Imputed** | 87.58 | 0.00 |
| **source_city_Imputed** | 1.30 | 0.25 |
| **source_country_Imputed** | 48.68 | 0.00 |
| **utm_source_Imputed** | 1.10 | 0.29 |
| **utm_medium_Imputed** | 128.24 | 0.00 |
| **des_city_Imputed** | 36.74 | 0.00 |
| **des_country_Imputed** | 48.01 | 0.00 |
| **movein_Imputed** | 124.59 | 0.00 |
| **room_type_Imputed** | 3.98 | 0.05 |

utm_source and source_city features dont have that discriminative power.

We decide to drop utm_source as it doesnt contribute much in prediction.

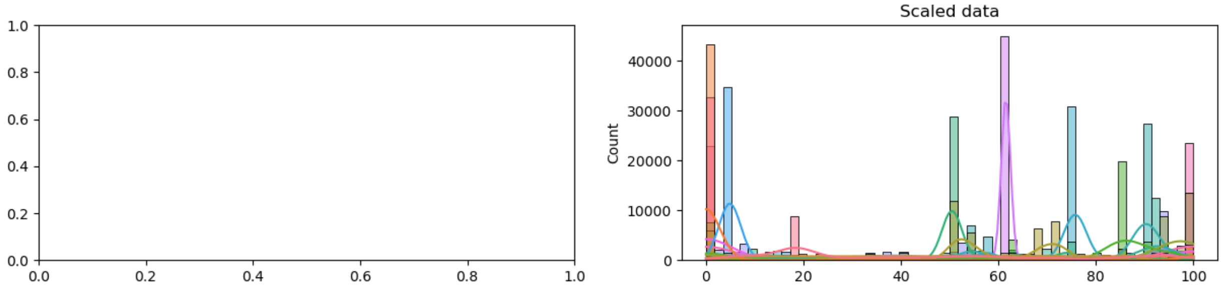
In [38]:
```
l4.drop(['utm_source_Imputed'], axis=1)
```

Out[38]:

| | Agent_id | OUTPUT | lead_id | lost_reason_Imputed | budget_Imputed | lease_Im |
|---|---|---|---|---|---|---|
| **0** | 10.084034 | 0.0 | 80.095113 | 72.413793 | 99.135135 | 85.8 |
| **1** | 16.806723 | 0.0 | 72.276156 | 55.172414 | 99.135135 | 85.8 |
| **2** | 73.109244 | 0.0 | 58.786487 | 72.413793 | 94.702703 | 85.8 |
| **3** | 92.436975 | 0.0 | 11.748114 | 55.172414 | 0.540541 | 0.6 |
| **4** | 10.084034 | 0.0 | 98.445392 | 27.586207 | 99.135135 | 85.8 |
| **...** | ... | ... | ... | ... | ... | |
| **46312** | 12.605042 | 0.0 | 10.806822 | 51.724138 | 99.135135 | 84.8 |
| **46313** | 20.168067 | 0.0 | 12.774680 | 82.758621 | 99.135135 | 92.2 |
| **46314** | 4.201681 | 0.0 | 85.067235 | 51.724138 | 97.837838 | 85.8 |
| **46315** | 50.420168 | 0.0 | 12.774680 | 51.724138 | 44.054054 | 30.0 |
| **46316** | 10.924370 | 0.0 | 45.788783 | 51.724138 | 96.918919 | 85.8 |

46317 rows × 14 columns

In [39]:
```
fig, ax = plt.subplots(1,2 ,figsize=(15, 3))
sns.histplot(scaled_data, ax=ax[1], kde=True, legend=False)
ax[1].set_title("Scaled data")
plt.show()
```




# MODEL SELECTION

In [40]:
```
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_recall_curve,classi
```

In [41]:
```
x_train,x_test,y_train,y_test = train_test_split(X,Y, test_size=0.2, rand
```

In [42]:
```
#Unbalanced data , we have to balance it
from imblearn.over_sampling import SMOTE
sm = SMOTE(sampling_strategy=0.75)
sm_x,sm_y = sm.fit_resample(x_train,y_train)
```

In [43]:
```
print(f"First we have the value counts:\n{y_train.value_counts()}\n\nAfte
```

```
First we have the value counts:
0.0      34595
100.0     2458
Name: OUTPUT, dtype: int64

After OverSampling now we have value counts:
0.0      34595
100.0    25946
Name: OUTPUT, dtype: int64
```

Data is now balanced

In [44]:
```
import os
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVR
from sklearn import linear_model
from sklearn.ensemble import RandomForestRegressor

arr = scaler.fit_transform(sm_x)
std_x = pd.DataFrame(arr, columns = sm_x.columns)
std_x.head()

arr1 = scaler.transform(x_test)
std_x_te = pd.DataFrame(arr1, columns = x_test.columns)
std_x_te.head()
```

Out[44]:

| | Agent_id | lead_id | lost_reason_Imputed | budget_Imputed | lease_Imputed | source_ |
|---|---|---|---|---|---|---|
| 0 | 83.050847 | 6.352903 | 62.068966 | 44.918919 | 63.548387 | 7 |
| 1 | 17.796610 | 86.054444 | 51.724138 | 56.486486 | 85.806452 | 5 |
| 2 | 55.932203 | 76.064283 | 89.655172 | 94.702703 | 85.806452 | 5 |
| 3 | 17.796610 | 54.945884 | 51.724138 | 99.135135 | 84.838710 | 5 |
| 4 | 11.864407 | 11.853067 | 55.172414 | 94.702703 | 84.838710 | 5 |

In [45]:
```
lst = [("LogisticRegression",LogisticRegression()),
       ("DecisionTree", DecisionTreeRegressor()),
       ("RandomForest",RandomForestRegressor())
       ]
for name,model in lst:
    model.fit(std_x,sm_y)
    y1 = model.predict(std_x)
    accuracy = accuracy_score(sm_y,y1)
    y2 = model.predict(std_x_te)
    acc_te = accuracy_score(y_test,y2)
    print(f"For {name}::\nThe Training Accuracy is: {accuracy}\nThe Testi
    print("--"*40)
```

```
For LogisticRegression::
The Training Accuracy is: 0.6995920120249087
The Testing Accuracy is: 0.7847582037996546
--------------------------------------------------------------------------
-------
For DecisionTree::
The Training Accuracy is: 1.0
The Testing Accuracy is: 0.9766839378238342
--------------------------------------------------------------------------
-------
For RandomForest::
The Training Accuracy is: 0.9092515815728184
The Testing Accuracy is: 0.9194732297063903
--------------------------------------------------------------------------
-------
```

In [46]:
```
#decision tree seems better on testing set.
```

# TRAINING AND METRICS

In [52]:
```
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(std_x,sm_y)
```

Out[52]:
```
DecisionTreeRegressor(random_state=0)
```

In [55]:
```
y_pr_train = regressor.predict(std_x)
acc_train = accuracy_score(sm_y,y_pr_train)
class_re = classification_report(sm_y,y_pr_train)
con_mat = confusion_matrix(sm_y,y_pr_train)
print("Confusion Matrix:\n",con_mat)
print("\n")
print("The accuracy of the model:",(acc_train)*100)
print("\n")
print("The classification report:\n",class_re)
```

```
Confusion Matrix:
 [[34595     0]
 [    0 25946]]


The accuracy of the model: 100.0


The classification report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     34595
       100.0       1.00      1.00      1.00     25946

    accuracy                           1.00     60541
   macro avg       1.00      1.00      1.00     60541
weighted avg       1.00      1.00      1.00     60541
```

# TESTING AND METRICS

In [56]:
```
y_pr_test = regressor.predict(std_x_te)
acc_test = accuracy_score(y_test,y_pr_test)
class_re1 = classification_report(y_test,y_pr_test)
con_mat1 = confusion_matrix(y_test,y_pr_test)
print("Confusion Matrix:\n",con_mat1)
print("\n")
print("The accuracy of the model:",(acc_test)*100)
print("\n")
print("The classification report:\n",class_re1)
```

```
Confusion Matrix:
 [[8527  122]
 [  91  524]]


The accuracy of the model: 97.70077720207254


The classification report:
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99      8649
       100.0       0.81      0.85      0.83       615

    accuracy                           0.98      9264
   macro avg       0.90      0.92      0.91      9264
weighted avg       0.98      0.98      0.98      9264
```

In [ ]:

In [ ]: