In [1]:
```
import random
import numpy as np
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from catboost import CatBoostClassifier

pd.set_option('display.max_columns', None)
```

In [2]:
```
fhr_transactions = pd.read_pickle('02fhr_fe1_output.pkl')
eval_transactions = pd.read_pickle('02eval_fe1_output.pkl')
```

As a rule, if the categorical predictor does not constitute 0.1% of the total population, we'll replace it with "OTHER"

In [3]:
```
categorical_predictors = fhr_transactions.select_dtypes(include=['object']).columns.tolist()
categorical_predictors_summary_df = pd.DataFrame(fhr_transactions[categorical_predictors].describe(in
clude='all')).transpose().reset_index(level = 0)
```

In [4]:
```
categorical_predictors_summary_df
```

Out[4]:

	index	count	unique	top	freq
0	trip_sta	504944	4	INVOICED	296335
1	mkt_cd	504944	23	US	247353
2	chan_type	504943	2	OFFLINE	449412
3	person_id	504944	179218	8b112f7a6b00250a034e5967c266ea7379f9065c061ed8...	494
4	dom_intl_in	504943	2	I	257494
5	city_nm	504944	563	NEW YORK	33528
6	state_cd	504944	165	CALIFORNIA	36416
7	ctry_cd	504944	123	UNITED STATES	193043
8	vend_cd	504944	1438	64086	11645
9	vend_brand	504944	129	[redacted]HOTELS AND RESORTS	59268
10	pwp_book_flag	504944	4	UNKNOWN	497055
11	prepay_in	504944	3	PAY LATER	299752
12	net_tkt_ct	504944	3	ACTIVE	405429
13	prog_id	504886	3	FHR	409768
14	card_type	492750	2	AX	492702
15	cm_dma	504944	209	nan	388298
16	card_ctgy	504944	188	PLATINUM	154199
17	srce_nm	504944	4	CC	296335
18	orig_state	504944	88	CALIFORNIA	52255
19	orig_ctry	504944	22	UNITED STATES	247353
20	pymt_form	504944	4	CC	473810
21	local_curr_cd	504944	61	USD	227404
22	doc_sta	504944	10	A	478873
23	rgn_cd	504944	4	USA	247353
24	acct_nm	504944	103	WNS PLATINUM	108321

	index	count	unique	top	freq
25	card_ctgy_grp	504944	8	PLATINUM	234163
26	htl_ctry_cd	504944	166	USA	185665
27	pseudo_city_cd	504944	68	Z8B0	200556
28	agcy_nm	504944	126	NOT-APPLICABLE	55531
29	acct_type	504944	131	PLATINUM	229062
30	cust_type	504944	9	PLATINUM	249564
31	trvl_ctry_orig_rgn	504944	5	NORTH AMERICA	280527
32	trvl_ctry_dest_rgn	504944	10	NORTH AMERICA	210727
33	seg_lvl_strt_month	504944	12	Mar	50029
34	cross_sell_type_A	504944	2	NO	435950
35	cross_sell_type_O	504944	2	NO	500691
36	cross_sell_type_S	504944	2	NO	503520
37	cross_sell_type_T	504944	2	NO	496724

In [5]:
```
cols_to_fill_w_mode = ['chan_type', 'dom_intl_in', 'prog_id', 'card_type', 'seg_lvl_end_month']
imp_mode = SimpleImputer(strategy='most_frequent')

fhr_transactions[cols_to_fill_w_mode] = imp_mode.fit_transform(fhr_transactions[cols_to_fill_w_mode])
eval_transactions[cols_to_fill_w_mode] = imp_mode.transform(eval_transactions[cols_to_fill_w_mode])
```

In [6]:
```
categorical_predictors_summary_df = pd.DataFrame(fhr_transactions[categorical_predictors].describe(in
clude='all')).transpose().reset_index(level = 0)
categorical_predictors_summary_df
```

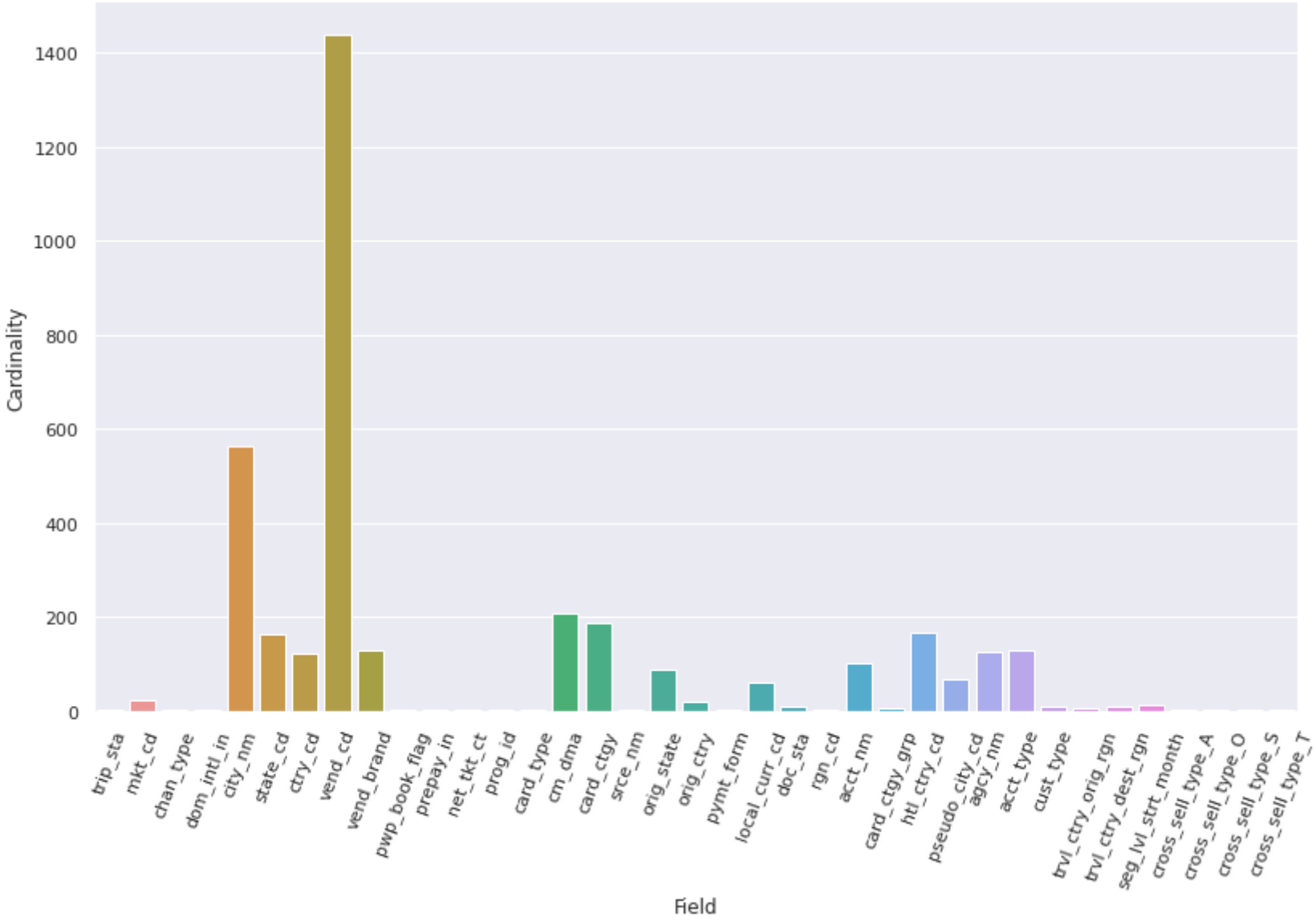
Out[6]:

	index	count	unique	top	freq
0	trip_sta	504944	4	INVOICED	296335
1	mkt_cd	504944	23	US	247353
2	chan_type	504944	2	OFFLINE	449413
3	person_id	504944	179218	8b112f7a6b00250a034e5967c266ea7379f9065c061ed8...	494
4	dom_intl_in	504944	2	I	257495
5	city_nm	504944	563	NEW YORK	33528
6	state_cd	504944	165	CALIFORNIA	36416
7	ctry_cd	504944	123	UNITED STATES	193043
8	vend_cd	504944	1438	64086	11645
9	vend_brand	504944	129	[redacted] HOTELS AND RESORTS	59268
10	pwp_book_flag	504944	4	UNKNOWN	497055
11	prepay_in	504944	3	PAY LATER	299752
12	net_tkt_ct	504944	3	ACTIVE	405429
13	prog_id	504944	3	FHR	409826
14	card_type	504944	2	AX	504896
15	cm_dma	504944	209	nan	388298
16	card_ctgy	504944	188	PLATINUM	154199
17	srce_nm	504944	4	CC	296335
18	orig_state	504944	88	CALIFORNIA	52255
19	orig_ctry	504944	22	UNITED STATES	247353
20	pymt_form	504944	4	CC	473810
21	local_curr_cd	504944	61	USD	227404
22	doc_sta	504944	10	A	478873
23	rgn_cd	504944	4	USA	247353
24	acct_nm	504944	103	WNS PLATINUM	108321

	index	count	unique	top	freq
25	card_ctgy_grp	504944	8	PLATINUM	234163
26	htl_ctry_cd	504944	166	USA	185665
27	pseudo_city_cd	504944	68	Z8B0	200556
28	agcy_nm	504944	126	NOT-APPLICABLE	55531
29	acct_type	504944	131	PLATINUM	229062
30	cust_type	504944	9	PLATINUM	249564
31	trvl_ctry_orig_rgn	504944	5	NORTH AMERICA	280527
32	trvl_ctry_dest_rgn	504944	10	NORTH AMERICA	210727
33	seg_lvl_strt_month	504944	12	Mar	50029
34	cross_sell_type_A	504944	2	NO	435950
35	cross_sell_type_O	504944	2	NO	500691
36	cross_sell_type_S	504944	2	NO	503520
37	cross_sell_type_T	504944	2	NO	496724

In [7]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [8]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
barplot = sns.barplot(x='index', y = 'unique', data = categorical_predictors_summary_df.drop(categori
cal_predictors_summary_df.index[3]))
barplot.set(xlabel='Field', ylabel='Cardinality')
plt.xticks(rotation=70)
plt.tight_layout()
```



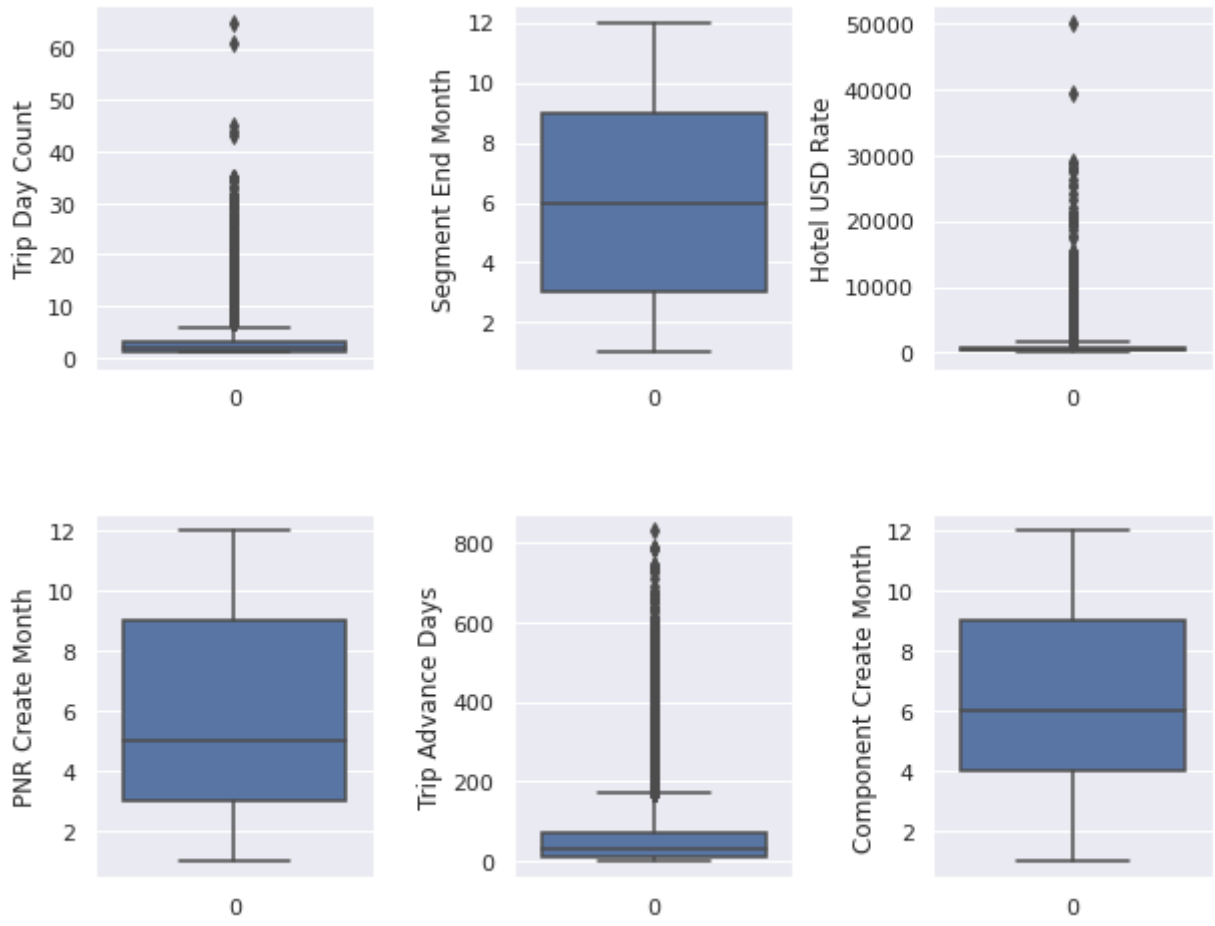
In [9]:
```
continous_predictors = fhr_transactions.select_dtypes(exclude=['object']).columns.tolist()
continous_predictors_summary_df = pd.DataFrame(fhr_transactions[continous_predictors].describe(includ
e='all')).transpose().reset_index(level = 0)
```

In [10]:
```
continous_predictors
```

Out[10]:
```
['trip_day_ct',
 'rm_no',
 'htl_usd_rt',
 'pnr_creat_ts_month',
 'trip_advance_days',
 'seg_lvl_end_month',
 'cmpnt_creat_dt_month']
```

In [11]:
```
fig, axs = plt.subplots(2,3, figsize=(10,8))
plt.subplots_adjust(wspace = 0.5, hspace = 0.4)
sns.boxplot(data=fhr_transactions['trip_day_ct'], ax = axs[0,0])
axs[0,0].set_ylabel("Trip Day Count")
sns.boxplot(data=fhr_transactions['seg_lvl_end_month'], ax = axs[0,1])
axs[0,1].set_ylabel("Segment End Month")
sns.boxplot(data=fhr_transactions['htl_usd_rt'], ax = axs[0,2])
axs[0,2].set_ylabel("Hotel USD Rate")
sns.boxplot(data=fhr_transactions['pnr_creat_ts_month'], ax = axs[1,0])
axs[1,0].set_ylabel("PNR Create Month")
sns.boxplot(data=fhr_transactions['trip_advance_days'], ax = axs[1,1])
axs[1,1].set_ylabel("Trip Advance Days")
sns.boxplot(data=fhr_transactions['cmpnt_creat_dt_month'], ax = axs[1,2])
axs[1,2].set_ylabel("Component Create Month")
```

Out[11]: Text(0, 0.5, 'Component Create Month')



In [12]:
```
continous_predictors_summary_df.index
```

Out[12]: RangeIndex(start=0, stop=7, step=1)

In [13]:
```
continous_predictors_summary_df
```

Out[13]:

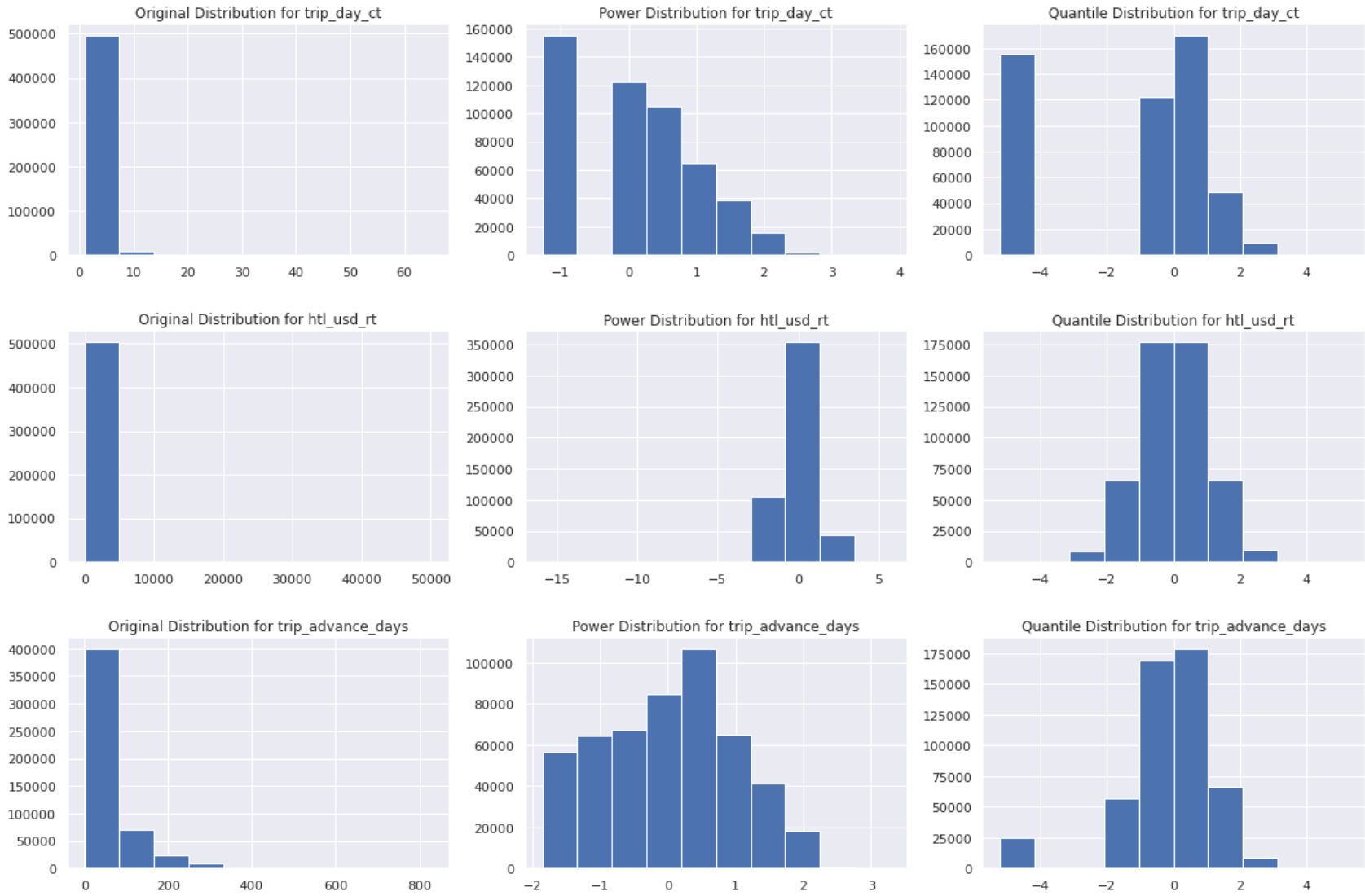
	index	count	mean	std	min	25%	50%	75%	max
0	trip_day_ct	504941.0	2.644139	1.777291	1.00	1.00	2.00	3.00	65.00
1	rm_no	504944.0	1.001984	0.052580	1.00	1.00	1.00	1.00	12.00
2	htl_usd_rt	504944.0	704.892498	595.436265	0.01	372.98	560.71	842.94	50146.21
3	pnr_creat_ts_month	504944.0	5.913018	3.525621	1.00	3.00	5.00	9.00	12.00
4	trip_advance_days	480383.0	54.101040	64.137750	1.00	9.00	30.00	74.00	829.00
5	seg_lvl_end_month	504944.0	6.311385	3.446270	1.00	3.00	6.00	9.00	12.00
6	cmpnt_creat_dt_month	504944.0	6.314607	3.301707	1.00	4.00	6.00	9.00	12.00

In [14]:
```
numeric_cols = ['trip_day_ct', 'trip_advance_days']
imp_mean = SimpleImputer(strategy='mean')
fhr_transactions[numeric_cols] = imp_mean.fit_transform(fhr_transactions[numeric_cols])
eval_transactions[numeric_cols] = imp_mean.transform(eval_transactions[numeric_cols])
```

In [15]:
```
transform_predictors = ['trip_day_ct', 'htl_usd_rt', 'trip_advance_days']
```

In [16]:
```
import matplotlib.pyplot as plt
from sklearn.preprocessing import PowerTransformer, QuantileTransformer

def test_transformers(columns):
    pt = PowerTransformer()
    qt = QuantileTransformer(n_quantiles=500, output_distribution='normal')
    fig = plt.figure(figsize=(20,30))
    j = 0
    for i in columns:
        plt.figure(figsize=(20,30))

        original_array = np.array(fhr_transactions[i]).reshape(-1, 1)
        j += 1
        plt.subplot(len(continous_predictors), 3, j)
        plt.hist(original_array)
        plt.title("Original Distribution for {}".format(i))

        pt_array = pt.fit_transform(original_array)
        j += 1
        plt.subplot(len(continous_predictors), 3, j)
        plt.hist(pt_array)
        plt.title("Power Distribution for {}".format(i))

        qt_array = qt.fit_transform(original_array)
        j += 1
        plt.subplot(len(continous_predictors), 3, j)
        plt.hist(qt_array)
        plt.title("Quantile Distribution for {}".format(i))
        plt.show()

test_transformers(transform_predictors)
```

```
<Figure size 1440x2160 with 0 Axes>
```



In [17]:
```
fhr_transactions.isna().sum()
```

Out[17]:

trip_sta	0
mkt_cd	0
chan_type	0
person_id	0
trip_day_ct	0
dom_intl_in	0
city_nm	0
state_cd	0
ctry_cd	0
vend_cd	0
vend_brand	0
pwp_book_flag	0
prepay_in	0
rm_no	0
net_tkt_ct	0
prog_id	0
card_type	0
cm_dma	0
card_ctgy	0
srce_nm	0
orig_state	0
orig_ctry	0
pymt_form	0
local_curr_cd	0
doc_sta	0
rgn_cd	0
acct_nm	0
card_ctgy_grp	0
htl_ctry_cd	0
pseudo_city_cd	0
agcy_nm	0
acct_type	0
cust_type	0
htl_usd_rt	0
trvl_ctry_orig_rgn	0
trvl_ctry_dest_rgn	0
pnr_creat_ts_month	0
seg_lvl_strt_month	0
trip_advance_days	0
seg_lvl_end_month	0
cross_sell_type_A	0
cross_sell_type_O	0
cross_sell_type_S	0

```
cross_sell_type_T       0
cmpnt_creat_dt_month    0
dtype: int64
```

In [18]:
```
print("Reducing cardinality by changing 0.1% to OTHER")

for index,row in categorical_predictors_summary_df.iterrows():
    cutoff_count = row['count']/(row['unique'] * 10)
    current_col = row['index']
    impacted_rows = fhr_transactions.groupby([current_col], dropna=False, as_index=False).filter(lamb
da x: len(x) < cutoff_count)
    print("For Predictor - {}/{} rows changed to OTHER".format(current_col, len(impacted_rows)))
    fhr_transactions.loc[impacted_rows.index, current_col] = 'OTHER'
```

```
Reducing cardinality by changing 0.1% to OTHER
For Predictor - trip_sta/6475 rows changed to OTHER
For Predictor - mkt_cd/3719 rows changed to OTHER
For Predictor - chan_type/0 rows changed to OTHER
For Predictor - person_id/0 rows changed to OTHER
For Predictor - dom_intl_in/0 rows changed to OTHER
For Predictor - city_nm/4802 rows changed to OTHER
For Predictor - state_cd/4988 rows changed to OTHER
For Predictor - ctry_cd/5518 rows changed to OTHER
For Predictor - vend_cd/5592 rows changed to OTHER
For Predictor - vend_brand/3679 rows changed to OTHER
For Predictor - pwp_book_flag/7889 rows changed to OTHER
For Predictor - prepay_in/5373 rows changed to OTHER
For Predictor - net_tkt_ct/13 rows changed to OTHER
For Predictor - prog_id/4597 rows changed to OTHER
For Predictor - card_type/48 rows changed to OTHER
For Predictor - cm_dma/9109 rows changed to OTHER
For Predictor - card_ctgy/4467 rows changed to OTHER
For Predictor - srce_nm/8790 rows changed to OTHER
For Predictor - orig_state/5855 rows changed to OTHER
For Predictor - orig_ctry/3719 rows changed to OTHER
For Predictor - pymt_form/10990 rows changed to OTHER
For Predictor - local_curr_cd/6075 rows changed to OTHER
For Predictor - doc_sta/5927 rows changed to OTHER
For Predictor - rgn_cd/0 rows changed to OTHER
For Predictor - acct_nm/3419 rows changed to OTHER
For Predictor - card_ctgy_grp/2071 rows changed to OTHER
For Predictor - htl_ctry_cd/5112 rows changed to OTHER
For Predictor - pseudo_city_cd/3381 rows changed to OTHER
For Predictor - agcy_nm/6031 rows changed to OTHER
For Predictor - acct_type/3501 rows changed to OTHER
For Predictor - cust_type/2918 rows changed to OTHER
For Predictor - trvl_ctry_orig_rgn/640 rows changed to OTHER
For Predictor - trvl_ctry_dest_rgn/3705 rows changed to OTHER
For Predictor - seg_lvl_strt_month/0 rows changed to OTHER
For Predictor - cross_sell_type_A/0 rows changed to OTHER
For Predictor - cross_sell_type_O/4253 rows changed to OTHER
For Predictor - cross_sell_type_S/1424 rows changed to OTHER
For Predictor - cross_sell_type_T/8220 rows changed to OTHER
```

In [19]:
```
fhr_transactions[categorical_predictors].describe(include='all').transpose()
```

Out[19]:

	count	unique	top	freq
trip_sta	504944	4	INVOICED	296335
mkt_cd	504944	19	US	247353
chan_type	504944	2	OFFLINE	449413
person_id	504944	179218	8b112f7a6b00250a034e5967c266ea7379f9065c061ed8...	494
dom_intl_in	504944	2	I	257495
city_nm	504944	267	NEW YORK	33528
state_cd	504944	97	CALIFORNIA	36416
ctry_cd	504944	65	UNITED STATES	193043
vend_cd	504944	903	64086	11645
vend_brand	504944	79	[redacted] HOTELS AND RESORTS	59268
pwp_book_flag	504944	2	UNKNOWN	497055
prepay_in	504944	3	PAY LATER	299752
net_tkt_ct	504944	3	ACTIVE	405429
prog_id	504944	3	FHR	409826
card_type	504944	2	AX	504896
cm_dma	504944	58	nan	388298
card_ctgy	504944	44	PLATINUM	154199
srce_nm	504944	3	CC	296335
orig_state	504944	57	CALIFORNIA	52255
orig_ctry	504944	18	UNITED STATES	247353
pymt_form	504944	3	CC	473810
local_curr_cd	504944	22	USD	227404
doc_sta	504944	3	A	478873
rgn_cd	504944	4	USA	247353
acct_nm	504944	38	WNS PLATINUM	108321

	count	unique	top	freq
card_ctgy_grp	504944	5	PLATINUM	234163
htl_ctry_cd	504944	71	USA	185665
pseudo_city_cd	504944	42	Z8B0	200556
agcy_nm	504944	70	NOT-APPLICABLE	55531
acct_type	504944	23	PLATINUM	229062
cust_type	504944	5	PLATINUM	249564
trvl_ctry_orig_rgn	504944	5	NORTH AMERICA	280527
trvl_ctry_dest_rgn	504944	8	NORTH AMERICA	210727
seg_lvl_strt_month	504944	12	Mar	50029
cross_sell_type_A	504944	2	NO	435950
cross_sell_type_O	504944	2	NO	500691
cross_sell_type_S	504944	2	NO	503520
cross_sell_type_T	504944	2	NO	496724

In [20]:
```
fhr_transactions['real_transaction'] = 1
eval_transactions['real_transaction'] = 1
```

In [21]:
```
hotel_specific_features = ['city_nm', 'state_cd', 'prog_id',
                           'ctry_cd', 'vend_cd', 'vend_brand', 'local_curr_cd',
                           'htl_ctry_cd', 'htl_usd_rt', 'trvl_ctry_dest_rgn']

randomized_hotel_only_features = fhr_transactions[hotel_specific_features].sample(frac=1, random_stat
e=21)

# Overwrite hotel features for negative cases
negative_cases = fhr_transactions.copy()
negative_cases[hotel_specific_features] = randomized_hotel_only_features.reset_index(drop=True)
negative_cases = negative_cases[fhr_transactions.columns]

negative_cases['real_transaction'] = 0
```

In [22]:
```
# Combine positive and negative cases into one dataset:
transaction_df = pd.concat([fhr_transactions, negative_cases])
```

In [23]:
```
# Replace domestic vs. international to be accurate to the newly-created trip
transaction_df['dom_intl_in'] = np.where(transaction_df['ctry_cd'] == transaction_df['orig_ctry'],
'D', 'I')
```

In [24]:
```
transaction_df.shape
```

Out[24]:
```
(1009888, 46)
```

In [25]:
```
transaction_df['real_transaction'].value_counts()
```

Out[25]:
```
0    504944
1    504944
Name: real_transaction, dtype: int64
```

In [26]:
```
transaction_df.to_pickle('03fhr_fe2_output.pkl')
eval_transactions.to_pickle('03eval_fe2_output.pkl')
```

In [ ]: