In [181]:
```
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 [182]:
```
import logging
logger = logging.getLogger()
logger.setLevel(logging.CRITICAL)
```

In [183]:
```
fhr_transactions = pd.read_pickle('01fhr_ede_output.pkl')
eval_transactions = pd.read_pickle('01eval_ede_output.pkl')
```

In [184]:
```
fhr_transactions.shape
```

Out[184]:
```
(504944, 62)
```

Field by Field Analysis

We'll collect the list of columns which should be dropped

In [185]:
```
drop_cols = list()
```

Field Info

In [186]:
```
fhr_transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 504944 entries, 0 to 504943
Data columns (total 62 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   trip_sta            504944 non-null  object
 1   mkt_cd              504944 non-null  object
 2   chan_type           504943 non-null  object
 3   pnr_creat_ts        504944 non-null  datetime64[ns]
 4   person_id           504944 non-null  object
 5   trvl_ct             305125 non-null  float64
 6   trip_strt_dt        504944 non-null  datetime64[ns]
 7   trip_end_dt         504944 non-null  datetime64[ns]
 8   trip_diff_day       504944 non-null  int64
 9   seg_lvl_strt_dt     504944 non-null  datetime64[ns]
 10  seg_lvl_end_dt      504941 non-null  datetime64[ns]
 11  trip_day_ct         504941 non-null  float64
 12  dest_airport_cd     495693 non-null  object
 13  dom_intl_in         493220 non-null  object
 14  city_nm             504748 non-null  object
 15  state_cd            197914 non-null  object
 16  ctry_cd             504742 non-null  object
 17  vend_cd             504944 non-null  object
 18  vend_nm             504944 non-null  object
 19  vend_brand          504590 non-null  object
 20  class_cd            8790 non-null    object
 21  class_srvc_cd       306617 non-null  object
 22  pymt_pct            2353 non-null    float64
 23  pwp_book_flag       7889 non-null    object
 24  mr_rdm_pt           4680 non-null    float64
 25  pkge_tax_am         8790 non-null    float64
 26  prepay_in           305125 non-null  object
 27  rm_no               504944 non-null  int64
 28  htl_rt              504944 non-null  float64
 29  trans_amt           504944 non-null  float64
 30  net_tkt_ct          305125 non-null  float64
 31  prog_id             504886 non-null  object
 32  card_type           492750 non-null  object
 33  cm_dma              504944 non-null  object
 34  cross_sell_type     504944 non-null  object
 35  card_ctgy           342059 non-null  object
 36  pax_seq_no          504944 non-null  int64
 37  srce_nm             504944 non-null  object
```

```
 38  orig_state          243292 non-null  object
 39  orig_ctry           504944 non-null  object
 40  fare_waive_am       5780 non-null    object
 41  pymt_form           484800 non-null  object
 42  local_curr_cd       504919 non-null  object
 43  doc_sta             484800 non-null  object
 44  cmpnt_creat_dt      504944 non-null  datetime64[ns]
 45  rgn_cd              504944 non-null  object
 46  acct_nm             496044 non-null  object
 47  htl_comm_in         296335 non-null  object
 48  htl_st_ad           504041 non-null  object
 49  card_ctgy_grp       310090 non-null  object
 50  htl_ctry_cd         495691 non-null  object
 51  pseudo_city_cd      504944 non-null  object
 52  agcy_nm             428060 non-null  object
 53  agcy_brnch_g6_cd    504944 non-null  object
 54  acct_type           469257 non-null  object
 55  cust_type           434980 non-null  object
 56  book_card_type      467551 non-null  object
 57  basic_supp_in       467551 non-null  object
 58  vend_no             461523 non-null  object
 59  htl_usd_rt          504940 non-null  float64
 60  trvl_ctry_orig_rgn  504944 non-null  object
 61  trvl_ctry_dest_rgn  504740 non-null  object
dtypes: datetime64[ns](6), float64(9), int64(3), object(44)
memory usage: 238.8+ MB
```

In [187]:
```
pd.set_option('display.max_rows', 500)
missing_cols_df = pd.DataFrame({'column_name': fhr_transactions.columns,
                                 'number_missing': fhr_transactions.isna().sum(),
                                 'percent_missing': fhr_transactions.isna().sum() * 100 / len(fhr_tra
nsactions)})
missing_cols_df
```

Out[187]:

	column_name	number_missing	percent_missing
trip_sta	trip_sta	0	0.000000
mkt_cd	mkt_cd	0	0.000000
chan_type	chan_type	1	0.000198
pnr_creat_ts	pnr_creat_ts	0	0.000000
person_id	person_id	0	0.000000
trvl_ct	trvl_ct	199819	39.572507
trip_strt_dt	trip_strt_dt	0	0.000000
trip_end_dt	trip_end_dt	0	0.000000
trip_diff_day	trip_diff_day	0	0.000000
seg_lvl_strt_dt	seg_lvl_strt_dt	0	0.000000
seg_lvl_end_dt	seg_lvl_end_dt	3	0.000594
trip_day_ct	trip_day_ct	3	0.000594
dest_airport_cd	dest_airport_cd	9251	1.832084
dom_intl_in	dom_intl_in	11724	2.321842
city_nm	city_nm	196	0.038816
state_cd	state_cd	307030	60.804763
ctry_cd	ctry_cd	202	0.040004
vend_cd	vend_cd	0	0.000000
vend_nm	vend_nm	0	0.000000
vend_brand	vend_brand	354	0.070107
class_cd	class_cd	496154	98.259213
class_srvc_cd	class_srvc_cd	198327	39.277029
pymt_pct	pymt_pct	502591	99.534008
pwp_book_flag	pwp_book_flag	497055	98.437649
mr_rdm_pt	mr_rdm_pt	500264	99.073165

	column_name	number_missing	percent_missing
pkge_tax_am	pkge_tax_am	496154	98.259213
prepay_in	prepay_in	199819	39.572507
rm_no	rm_no	0	0.000000
htl_rt	htl_rt	0	0.000000
trans_amt	trans_amt	0	0.000000
net_tkt_ct	net_tkt_ct	199819	39.572507
prog_id	prog_id	58	0.011486
card_type	card_type	12194	2.414921
cm_dma	cm_dma	0	0.000000
cross_sell_type	cross_sell_type	0	0.000000
card_ctgy	card_ctgy	162885	32.258033
pax_seq_no	pax_seq_no	0	0.000000
srce_nm	srce_nm	0	0.000000
orig_state	orig_state	261652	51.818023
orig_ctry	orig_ctry	0	0.000000
fare_waive_am	fare_waive_am	499164	98.855319
pymt_form	pymt_form	20144	3.989353
local_curr_cd	local_curr_cd	25	0.004951
doc_sta	doc_sta	20144	3.989353
cmpnt_creat_dt	cmpnt_creat_dt	0	0.000000
rgn_cd	rgn_cd	0	0.000000
acct_nm	acct_nm	8900	1.762572
htl_comm_in	htl_comm_in	208609	41.313294
htl_st_ad	htl_st_ad	903	0.178832
card_ctgy_grp	card_ctgy_grp	194854	38.589230
htl_ctry_cd	htl_ctry_cd	9253	1.832480

	column_name	number_missing	percent_missing
pseudo_city_cd	pseudo_city_cd	0	0.000000
agcy_nm	agcy_nm	76884	15.226243
agcy_brnch_g6_cd	agcy_brnch_g6_cd	0	0.000000
acct_type	acct_type	35687	7.067516
cust_type	cust_type	69964	13.855794
book_card_type	book_card_type	37393	7.405376
basic_supp_in	basic_supp_in	37393	7.405376
vend_no	vend_no	43421	8.599171
htl_usd_rt	htl_usd_rt	4	0.000792
trvl_ctry_orig_rgn	trvl_ctry_orig_rgn	0	0.000000
trvl_ctry_dest_rgn	trvl_ctry_dest_rgn	204	0.040401

Find related columns by name

Let's also sort by the name to identify which fields may be related to each other - by field names

In [188]:
```
missing_cols_df.sort_values('column_name')
```

Out[188]:

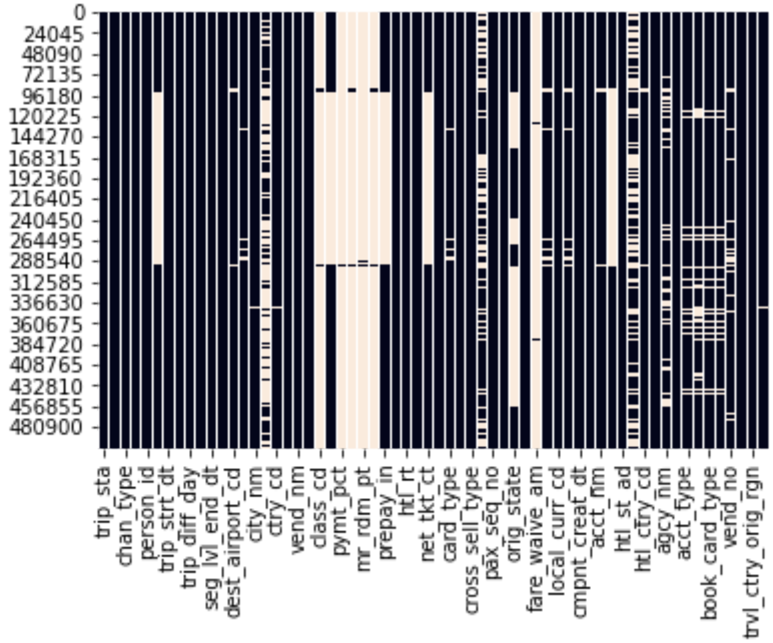
	column_name	number_missing	percent_missing
acct_nm	acct_nm	8900	1.762572
acct_type	acct_type	35687	7.067516
agcy_brnch_g6_cd	agcy_brnch_g6_cd	0	0.000000
agcy_nm	agcy_nm	76884	15.226243
basic_supp_in	basic_supp_in	37393	7.405376
book_card_type	book_card_type	37393	7.405376
card_ctgy	card_ctgy	162885	32.258033
card_ctgy_grp	card_ctgy_grp	194854	38.589230
card_type	card_type	12194	2.414921
chan_type	chan_type	1	0.000198
city_nm	city_nm	196	0.038816
class_cd	class_cd	496154	98.259213
class_srvc_cd	class_srvc_cd	198327	39.277029
cm_dma	cm_dma	0	0.000000
cmpnt_creat_dt	cmpnt_creat_dt	0	0.000000
cross_sell_type	cross_sell_type	0	0.000000
ctry_cd	ctry_cd	202	0.040004
cust_type	cust_type	69964	13.855794
dest_airport_cd	dest_airport_cd	9251	1.832084
doc_sta	doc_sta	20144	3.989353
dom_intl_in	dom_intl_in	11724	2.321842
fare_waive_am	fare_waive_am	499164	98.855319
htl_comm_in	htl_comm_in	208609	41.313294
htl_ctry_cd	htl_ctry_cd	9253	1.832480
htl_rt	htl_rt	0	0.000000

	column_name	number_missing	percent_missing
htl_st_ad	htl_st_ad	903	0.178832
htl_usd_rt	htl_usd_rt	4	0.000792
local_curr_cd	local_curr_cd	25	0.004951
mkt_cd	mkt_cd	0	0.000000
mr_rdm_pt	mr_rdm_pt	500264	99.073165
net_tkt_ct	net_tkt_ct	199819	39.572507
orig_ctry	orig_ctry	0	0.000000
orig_state	orig_state	261652	51.818023
pax_seq_no	pax_seq_no	0	0.000000
person_id	person_id	0	0.000000
pkge_tax_am	pkge_tax_am	496154	98.259213
pnr_creat_ts	pnr_creat_ts	0	0.000000
prepay_in	prepay_in	199819	39.572507
prog_id	prog_id	58	0.011486
pseudo_city_cd	pseudo_city_cd	0	0.000000
pwp_book_flag	pwp_book_flag	497055	98.437649
pymt_form	pymt_form	20144	3.989353
pymt_pct	pymt_pct	502591	99.534008
rgn_cd	rgn_cd	0	0.000000
rm_no	rm_no	0	0.000000
seg_lvl_end_dt	seg_lvl_end_dt	3	0.000594
seg_lvl_strt_dt	seg_lvl_strt_dt	0	0.000000
srce_nm	srce_nm	0	0.000000
state_cd	state_cd	307030	60.804763
trans_amt	trans_amt	0	0.000000
trip_day_ct	trip_day_ct	3	0.000594

	column_name	number_missing	percent_missing
trip_diff_day	trip_diff_day	0	0.000000
trip_end_dt	trip_end_dt	0	0.000000
trip_sta	trip_sta	0	0.000000
trip_strt_dt	trip_strt_dt	0	0.000000
trvl_ct	trvl_ct	199819	39.572507
trvl_ctry_dest_rgn	trvl_ctry_dest_rgn	204	0.040401
trvl_ctry_orig_rgn	trvl_ctry_orig_rgn	0	0.000000
vend_brand	vend_brand	354	0.070107
vend_cd	vend_cd	0	0.000000
vend_nm	vend_nm	0	0.000000
vend_no	vend_no	43421	8.599171

Now, we see how the fields are related to each other. For example - now we can clearly see that all fields which begin with vend_ are all vendor related params.

In [189]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(fhr_transactions.isnull(), cbar = False)
plt.show()
```



Feature by Feature Analysis

trip_sta - Status of the trip

In [190]:
```
fhr_transactions.groupby(['trip_sta'], dropna=False, as_index=False).size()
```

Out[190]:

	trip_sta	size
0	ACTIVE	145452
1	CANCELLED	56682
2	INVOICED	296335
3	SUCCESS	6475

trip_sta has no nulls and it is good.

mkt_cd - Market Code

In [191]:
```
fhr_transactions.groupby(['mkt_cd'], dropna=False, as_index=False).size()
```

Out[191]:

	mkt_cd	size
0	AR	640
1	AT	1743
2	AU	23072
3	CA	20327
4	DE	23525
5	ES	2476
6	FI	1007
7	FR	6381
8	HK	28106
9	ICC	13101
10	IN	11670
11	IT	8275
12	JP	34099
13	MX	12847
14	NL	3505
15	NO	54
16	NZ	275
17	SE	2702
18	SG	7592
19	TH	4460
20	TW	18450
21	UK	33284
22	US	247353

mkt_cd has no nulls. So we should be good.

chan_type - Channel Type

In [192]:
```
fhr_transactions.groupby(['chan_type'], dropna=False, as_index=False).size()
```

Out[192]:

	chan_type	size
0	OFFLINE	449412
1	ONLINE	55531
2	NaN	1

In [193]:
```
fhr_transactions[fhr_transactions['chan_type'].isna()]
```

Out[193]:

	trip_sta	mkt_cd	chan_type	pnr_creat_ts		person_id	trvl_ct	trip_strt_dt	trip_end_(
271342	CANCELLED	US	NaN	2019-05-02	3c44f36d77404a86254a718a75991d1361523fdea0bff7...	NaN	2018-08-06	2019-05-2	

There is one NaN. We'll impute this with most frequent value.

pnr_creat_ts - Timestamp when the ticket was created

In [194]:
```
fhr_transactions['pnr_creat_ts'].value_counts()
```

Out[194]:
```
2019-01-14 00:00:00    1345
2019-01-07 00:00:00    1334
2019-01-15 00:00:00    1290
2019-01-29 00:00:00    1277
2019-01-08 00:00:00    1269
                       ... 
2019-07-27 23:00:00       1
2018-11-16 16:00:00       1
2019-05-17 07:00:00       1
2019-09-09 20:00:00       1
2019-12-10 12:00:00       1
Name: pnr_creat_ts, Length: 6025, dtype: int64
```

We see that this column has high cardinality.

Lets split the date/time month and ignore this field. #dropthis

In [195]:
```
fhr_transactions['pnr_creat_ts'].dt.month.value_counts()
```

Out[195]:
```
1     58906
3     53098
2     51231
4     49245
5     43133
10    38467
9     35924
7     35736
12    35673
8     35668
11    34740
6     33123
Name: pnr_creat_ts, dtype: int64
```

We see that the dataset is very balanced and it has data for all the months.

In [196]:
```
fhr_transactions['pnr_creat_ts_month'] = fhr_transactions['pnr_creat_ts'].dt.month
eval_transactions['pnr_creat_ts_month'] = eval_transactions['pnr_creat_ts'].dt.month
```

In [197]:
```
drop_cols.append('pnr_creat_ts')
```

person_id - Unique identifier for the Person

In [198]:
```
import hashlib
# fhr_transactions['person_id'] = fhr_transactions['person_id'].astype(str).apply(lambda x: hashlib.s
ha256(x.encode()).hexdigest())
# eval_transactions['person_id'] = eval_transactions['person_id'].astype(str).apply(lambda x: hashli
b.sha256(x.encode()).hexdigest())
```

trvl_ct - The number of travellers in the trip

In [199]:
```
fhr_transactions.groupby(['trvl_ct'], dropna=False, as_index=False).size()
```

Out[199]:

	trvl_ct	size
0	1.0	59276
1	2.0	235476
2	3.0	8106
3	4.0	1935
4	5.0	181
5	6.0	116
6	7.0	14
7	8.0	17
8	9.0	4
9	NaN	199819

In [202]:
```
fhr_transactions['trvl_ct'].isna().sum()
```
Out[202]:
```
199819
```

1. We see there are a lot of NA's. May need imputation.
2. Also, we see that the srce_nm is [redacted] for all those which have trvl_ct as NA

In [203]:
```
fhr_transactions.groupby(['trvl_ct', 'rm_no'], dropna=False, as_index=False).size()
```

Out[203]:

	trvl_ct	rm_no	size
0	1.0	1	58722
1	1.0	2	502
2	1.0	3	40
3	1.0	4	9
4	1.0	6	2
5	1.0	12	1
6	2.0	1	235332
7	2.0	2	117
8	2.0	3	26
9	2.0	4	1
10	3.0	1	8104
11	3.0	2	1
12	3.0	3	1
13	4.0	1	1935
14	5.0	1	181
15	6.0	1	116
16	7.0	1	14
17	8.0	1	17
18	9.0	1	4
19	NaN	1	199645
20	NaN	2	154
21	NaN	3	17
22	NaN	4	3

When we look at this, it looks like the column trvl_ct is not very reliable.

The number of rooms booked and the number of traveler counts does not match.

We may drop this column. #dropthis #[redacted]

In [204]:
```
drop_cols.append('trvl_ct')
```

trip_strt_dt. Trip Start Date

There are no null values in this. As such, this predictor does not add any value.

In [205]:
```
fhr_transactions['trip_strt_month'] = fhr_transactions['trip_strt_dt'].dt.month
eval_transactions['trip_strt_month'] = eval_transactions['trip_strt_dt'].dt.month
```

Again, we can extract just the month from trip start month and drop the day and the year. #dropthis

In [206]:
```
drop_cols.append('trip_strt_dt')
```

trip_end_dt. Trip End Date

Again, we can ignore the date and the year and consider only the month. #dropthis

In [207]:
```
fhr_transactions['trip_end_month'] = fhr_transactions['trip_end_dt'].dt.month
eval_transactions['trip_end_month'] = eval_transactions['trip_end_dt'].dt.month
```

In [208]:
```
drop_cols.append('trip_end_dt')
```

trip_diff_day Total days in trip

In [209]:
```
fhr_transactions.groupby(['trip_diff_day'], dropna=False, as_index=False).size()
```

Out[209]:

	trip_diff_day	size
0	1	113864
1	2	92295
2	3	83600
3	4	59843
4	5	30764
...	...	...
1348	1787	1
1349	1890	1
1350	1937	1
1351	1962	1
1352	737192	1

1353 rows × 2 columns

This field does not have lot of valid values. We should drop this in favor of trip_day_ct which is the date difference between segments #dropthis

In [210]:
```
drop_cols.append('trip_diff_day')
```

seg_lvl_strt_dt Segment Start Date

In [211]:
```
fhr_transactions['seg_lvl_strt_month'] = fhr_transactions['seg_lvl_strt_dt'].dt.month
eval_transactions['seg_lvl_strt_month'] = eval_transactions['seg_lvl_strt_dt'].dt.month
```

Lets do the same. Lets drop the date and the year. #dropthis

The difference between this and the PNR create date can show the number of days in advance the ticket was booked.

In [212]:
```
drop_cols.append('seg_lvl_strt_dt')
```

In [213]:
```
fhr_transactions[['seg_lvl_strt_dt','pnr_creat_ts']] = fhr_transactions[['seg_lvl_strt_dt','pnr_creat_ts']].apply(pd.to_datetime)
fhr_transactions['trip_advance_days'] = (fhr_transactions['seg_lvl_strt_dt'] - fhr_transactions['pnr_creat_ts']).dt.days

eval_transactions[['seg_lvl_strt_dt','pnr_creat_ts']] = eval_transactions[['seg_lvl_strt_dt','pnr_creat_ts']].apply(pd.to_datetime)
eval_transactions['trip_advance_days'] = (eval_transactions['seg_lvl_strt_dt'] - eval_transactions['pnr_creat_ts']).dt.days
```

In [214]:
```
fhr_transactions.groupby(['trip_advance_days'], dropna=False, as_index=False).size()
```

Out[214]:

	trip_advance_days	size
0	-381	1
1	-372	2
2	-107	1
3	-56	1
4	-48	1
...	...	...
509	734	3
510	738	9
511	739	3
512	787	4
513	829	1

514 rows × 2 columns

We'll consider all the rows which have negative number of days as NAs.

In [215]:
```
fhr_transactions['trip_advance_days'] = fhr_transactions['trip_advance_days'].apply(lambda x: np.NaN
if x <= 0 else x)
eval_transactions['trip_advance_days'] = eval_transactions['trip_advance_days'].apply(lambda x: np.Na
N if x <= 0 else x)
```

In [216]:
```
fhr_transactions.groupby(['trip_advance_days'], dropna=False, as_index=False).size()
```

Out[216]:

	trip_advance_days	size
0	1.0	24616
1	2.0	16923
2	3.0	14746
3	4.0	13344
4	5.0	11685
...	...	...
496	738.0	9
497	739.0	3
498	787.0	4
499	829.0	1
500	NaN	24561

501 rows × 2 columns

IMPUTETHIS

seg_lvl_end_dt Segment End Date

In [217]:
```
fhr_transactions['seg_lvl_end_month'] = fhr_transactions['seg_lvl_end_dt'].dt.month
eval_transactions['seg_lvl_end_month'] = eval_transactions['seg_lvl_end_dt'].dt.month
```

Lets do the same. Lets drop the date and the year. #dropthis

In [218]:
```
drop_cols.append('seg_lvl_end_dt')
```

trip_day_ct - No of days of the trip

In [219]:
```
fhr_transactions.groupby(['trip_day_ct'], dropna=False, as_index=False).size().head()
```

Out[219]:

	trip_day_ct	size
0	1.0	155187
1	2.0	122639
2	3.0	104774
3	4.0	65077
4	5.0	26182

There are only 3 NaNs. We should be good with this.

dest_airport_cd Destination airport code

This field is required to identify the destination of the travel. We have the following other fields which can give information about the destination.

1. city_nm
2. ctry_cd
3. dest_airport_cd
4. dom_intl_in
5. htl_ctry_cd
6. htl_st_ad
7. htl_usd_rt
8. local_curr_cd
9. pseudo_city_cd
10. state_cd
11. trvl_ctry_dest_rgn

This field tells us 2 things.

1. The city of the destination. We'll see if we can combine this with city_nm (which has less nulls) and come up with a combined column.
2. This iternary has a Air travel. We'll see if we can combine this with cross_sell_type (which has the travel type).

In [220]:
```
fhr_transactions['dest_airport_cd'].isna().sum()
```
Out[220]:
```
9251
```

In [221]:
```
drop_cols.append('dest_airport_cd')
```

dom_intl_in Domestic or international Indicator

This is a critical field since it can be used to identify both source and destination - if they are not present. We will use this in-combination with the trip origin and trip destination fields.

In [222]:
```
fhr_transactions['dom_intl_in'].isna().sum()
```
Out[222]:
```
11724
```

In [223]:
```
fhr_transactions.groupby(['dom_intl_in'], dropna=False, as_index=False).size()
```

Out[223]:

	dom_intl_in	size
0	D	242122
1	I	250722
2	T	376
3	NaN	11724

For all the records where we have OTHER, lets see if we can populate it based on the Origin and Destination country codes.

In [224]:
```
def get_dom_intl_in(row):
    if (pd.notnull(row['dom_intl_in']) & (row['dom_intl_in'] != 'T')):
        return row['dom_intl_in']

    if (pd.isna(row['ctry_cd']) | pd.isna(row['orig_ctry'])):
        return row['dom_intl_in']

    if row['orig_ctry'] == row['ctry_cd']:
        return 'D'
    else:
        return 'I'

fhr_transactions['dom_intl_in'] = fhr_transactions.apply(lambda row: get_dom_intl_in(row), axis = 1)
eval_transactions['dom_intl_in'] = eval_transactions.apply(lambda row: get_dom_intl_in(row), axis = 1
)
```

In [225]:
```
fhr_transactions.groupby(['dom_intl_in'], dropna=False, as_index=False).size()
```

Out[225]:

	dom_intl_in	size
0	D	247449
1	I	257494
2	NaN	1

city_nm City name of the destination

In [226]:
```
fhr_transactions['city_nm'].isna().sum()
```
Out[226]:
```
196
```

Lets combine city_nm and dest_airport_cd to get the destination city name.

In [227]:
```
def get_city_nm(row):
    if pd.isnull(row['city_nm']):
        return row['dest_airport_cd']
    else:
        return row['city_nm']

fhr_transactions['city_nm'] = fhr_transactions.apply(lambda row: get_city_nm(row), axis = 1)
eval_transactions['city_nm'] = eval_transactions.apply(lambda row: get_city_nm(row), axis = 1)
```

In [228]:
```
fhr_transactions['city_nm'].isna().sum()
```
Out[228]:
```
0
```

state_cd State code of the destination

In [229]:
```
fhr_transactions['state_cd'].isna().sum()
```
Out[229]:
```
307030
```

In [230]:
```
fhr_transactions[fhr_transactions['state_cd'].isna()].groupby(['ctry_cd'], dropna=False, as_index=False).size().head()
```

Out[230]:

	ctry_cd	size
0	ANDORRA	2
1	ANGOLA	442
2	ANGUILLA	561
3	ANTIGUA AND BARBUDA	145
4	ARGENTINA	1891

We see that the state code is missing for mostly outside US destination. We can reuse the country code into this field for non US countries.

ctry_cd Country Code of the destination

For all the rows which have OTHER, lets see if we can populate it based on the Domestic/International Indicator. If it is domestic, the origin country and the destination country should be the same.

In [231]:
```
fhr_transactions['ctry_cd'].isna().sum()
```

Out[231]: 202

In [232]:
```
def get_ctry_cd(row):
    if pd.notnull(row['ctry_cd']):
        return row['ctry_cd']

    if row['dom_intl_in'] == 'D':
        return row['orig_ctry']

    if pd.notnull(row['local_curr_cd']):
        return row['local_curr_cd']

    return np.NaN

fhr_transactions['ctry_cd'] = fhr_transactions.apply(lambda row: get_ctry_cd(row), axis = 1)
eval_transactions['ctry_cd'] = eval_transactions.apply(lambda row: get_ctry_cd(row), axis = 1)
```

Lets fix state_cd based on ctry_cd and local currency code

In [233]:
```
def get_state_cd(row):
    if pd.notnull(row['state_cd']):
        return row['state_cd']
    return row['ctry_cd']

fhr_transactions['state_cd'] = fhr_transactions.apply(lambda row: get_state_cd(row), axis = 1)
eval_transactions['state_cd'] = eval_transactions.apply(lambda row: get_state_cd(row), axis = 1)
```

In [234]:
```
fhr_transactions['state_cd'].isna().sum()
```

Out[234]: 0

vend_cd ID of the Vendor

In [235]:
```
fhr_transactions['vend_cd'].isna().sum()
```

Out[235]: 0

vend_nm Vendor Name

In [236]:
```
fhr_transactions['vend_nm'].isna().sum()
```
Out[236]: 0

We can drop this because it does not add any value. #dropthis

In [237]:
```
drop_cols.append('vend_nm')
```

vend_brand Vendor Brand

In [238]:
```
fhr_transactions['vend_brand'].isna().sum()
```
Out[238]: 354

In [239]:
```
fhr_transactions[fhr_transactions['vend_brand'].isna()].groupby(['vend_nm', 'vend_brand'], dropna=False, as_index=False).size()
```
Out[239]:

	vend_nm	vend_brand	size
0	CONRAD DUBLIN	NaN	1
1	FOUR SEASONS RESORT - NEVIS	NaN	2
2	FOUR SEASONS RESORT MAUI AT WAILEA	NaN	32
3	HOTEL CONTINENTAL	NaN	280
4	PARK HYATT - CHICAGO	NaN	6
5	PARK HYATT CHICAGO	NaN	2
6	THE RITZ-CARLTON, AMELIA ISLAND	NaN	1
7	THE TOWERS AT LOTTE NEW YORK PALACE	NaN	1
8	TRUMP INTERNATIONAL HOTEL & TOWER NEW YORK	NaN	29

Lets use vend_cd as a proxy for vend_brand.

In [240]:
```
fhr_transactions['vend_brand'] = fhr_transactions.apply(lambda x: x['vend_brand'] if pd.notnull(x['ve
nd_brand']) else x['vend_cd'], axis = 1)
eval_transactions['vend_brand'] = eval_transactions.apply(lambda x: x['vend_brand'] if pd.notnull(x[
'vend_brand']) else x['vend_cd'], axis = 1)
```

class_cd Class Code

In [241]:
```
fhr_transactions[fhr_transactions['class_cd'].isna()].groupby(['srce_nm'], dropna=False, as_index=Fal
se).size()
```

Out[241]:

	srce_nm	size
0	CC	296335
1	[redacted]	199819

This field is missing for most of the rows. Can we drop this column? #dropthis

In [242]:
```
drop_cols.append('class_cd')
```

class_srvc_cd Class Service Code

In [243]:
```
fhr_transactions[fhr_transactions['class_srvc_cd'].isna()].groupby(['srce_nm'], dropna=False, as_inde
x=False).size()
```

Out[243]:

	srce_nm	size
0	[redacted]	73
1	[redacted]	198254

Looks like for tickets booked thru [redacted], we dont have this. #[redacted] #dropthis

In [244]:
```
drop_cols.append('class_srvc_cd')
```

pymt_pct Payment percent

In [245]:
```
fhr_transactions['pymt_pct'].isna().sum()
```

Out[245]:
```
502591
```

We see its missing for 99.5% of the rows. We can drop this column. #dropthis

In [246]:
```
drop_cols.append('pymt_pct')
```

pwp_book_flag - If the booking was made using points.

In [247]:
```
# pwp_book_flag
fhr_transactions.groupby(['pwp_book_flag'], dropna=False, as_index=False).size()
```

Out[247]:

	pwp_book_flag	size
0	FULL	1186
1	NONE	5536
2	PARTIAL	1167
3	NaN	497055

In [248]:
```
fhr_transactions['pwp_book_flag'] = fhr_transactions.apply(lambda x: "UNKNOWN" if pd.isnull(x['pwp_bo
ok_flag']) else x['pwp_book_flag'], axis = 1)
eval_transactions['pwp_book_flag'] = eval_transactions.apply(lambda x: "UNKNOWN" if pd.isnull(x['pwp_
book_flag']) else x['pwp_book_flag'], axis = 1)
```

We can derive more details about this flag when we check the amounts.

mr_rdm_pt

We see its missing for 99.5% of the rows. We can drop this column. #dropthis

In [249]:
```
# mr_rdm_pt
fhr_transactions['pwp_book_flag'].isna().sum()
```

Out[249]: 0

In [250]:
```
drop_cols.append('mr_rdm_pt')
```

pkge_tax_am Package tax amount

In [251]:
```
fhr_transactions['pkge_tax_am'].isna().sum()
```

Out[251]: 496154

This is missing for 98% of the rows. We can drop this in favor of other amounts. #dropthis

In [252]:
```
drop_cols.append('pkge_tax_am')
```

prepay_in Prepayment Indicator

In [253]:
```
# prepay_in
fhr_transactions.groupby(['prepay_in'], dropna=False, as_index=False).size()
```

Out[253]:

	prepay_in	size
0	PAY LATER	299752
1	PAY NOW	5373
2	NaN	199819

In [254]:
```
fhr_transactions.groupby(['srce_nm', 'chan_type', 'pwp_book_flag', 'prepay_in'], dropna=False, as_ind
ex=False).size()
```

Out[254]:

	srce_nm	chan_type	pwp_book_flag	prepay_in	size
0	CC	OFFLINE	UNKNOWN	PAY LATER	269893
1	CC	ONLINE	UNKNOWN	PAY LATER	26442
2	[redacted]	ONLINE	FULL	PAY NOW	1186
3	[redacted]	ONLINE	NONE	PAY LATER	2610
4	[redacted]	ONLINE	NONE	PAY NOW	2926
5	[redacted]	ONLINE	PARTIAL	PAY NOW	1167
6	[redacted]	ONLINE	UNKNOWN	PAY LATER	807
7	[redacted]	ONLINE	UNKNOWN	PAY NOW	94
8	[redacted]	OFFLINE	UNKNOWN	NaN	179519
9	[redacted]	ONLINE	UNKNOWN	NaN	20299
10	[redacted]	NaN	UNKNOWN	NaN	1

We observe the following.

1. For all booking thru [redacted], there is not points redemtion and prepay indicator available. Since it is not thru Central Command, we can assume that it is NONE for pwp_book_flag AND OTHER (which means Pay Now) for prepay_in.

SABRE

In [255]:
```
fhr_transactions['prepay_in'] = fhr_transactions['prepay_in'].apply(lambda x: "UNKNOWN" if pd.isna(x)
else x)
```

rm_no - No of Rooms

In [256]:
```
fhr_transactions.groupby(['rm_no'], dropna=False, as_index=False).size()
```

Out[256]:

	rm_no	size
0	1	504070
1	2	774
2	3	84
3	4	13
4	6	2
5	12	1

htl_rt - Nightly rate at the hotel

In [257]:
```
fhr_transactions['htl_rt'].isna().sum()
```

Out[257]: 0

In [258]:
```
fhr_transactions['htl_rt'] = fhr_transactions['htl_rt'].apply(lambda x: np.NaN if x <= 0 else x)
eval_transactions['htl_rt'] = eval_transactions['htl_rt'].apply(lambda x: np.NaN if x <= 0 else x)
```

Lets impute this field using linear regression

In [259]:
```
fhr_transactions['htl_rt'] = fhr_transactions['htl_rt'].interpolate(method='linear', limit_direction='both', axis=0)
eval_transactions['htl_rt'] = eval_transactions['htl_rt'].interpolate(method='linear', limit_direction='both', axis=0)
```

Also, based on the Exploratory Data Analysis we did, this is normally distributed.

We should ignore this field since it is not adjusted per FX rates. Use htl_usd_rt instead.

In [260]:
```
drop_cols.append('htl_rt')
```

trans_amt - Transaction Amount

In [261]:
```
fhr_transactions[['person_id']][fhr_transactions['trans_amt'].isna() | fhr_transactions['trans_amt'] <= 0].count()
```

Out[261]:
```
person_id    46
dtype: int64
```

We should ignore this field since it is not adjusted per FX rates. Use htl_usd_rt instead.

In [262]:
```
drop_cols.append('trans_amt')
```

net_tkt_ct - Indicates if transaction is active or not

In [263]:
```
## net_tkt_ct
fhr_transactions.groupby(['trip_sta', 'prepay_in', 'net_tkt_ct'], dropna=False, as_index=False).size
()
```

Out[263]:

	trip_sta	prepay_in	net_tkt_ct	size
0	ACTIVE	UNKNOWN	NaN	145452
1	CANCELLED	PAY LATER	1.0	1310
2	CANCELLED	PAY NOW	1.0	1005
3	CANCELLED	UNKNOWN	NaN	54367
4	INVOICED	PAY LATER	-1.0	13
5	INVOICED	PAY LATER	0.0	45135
6	INVOICED	PAY LATER	1.0	251187
7	SUCCESS	PAY LATER	1.0	2107
8	SUCCESS	PAY NOW	1.0	4368

Based on trip_sta, we can adjust net_tkt_ct. Also, this is a categorical predictor. Lets consider 3 values.

1. Instead of 1, put it as 'ACTIVE'
2. Instead of '-1', put it as 'REFUND'
3. Instead of 0, put 'NO-REFUND'

In [264]:
```
def get_net_tkt_ct(row):
    if pd.isnull(row['net_tkt_ct']):
        if (row['trip_sta'] == 'ACTIVE'):
            return 'ACTIVE'
        else:
            return 'NO-REFUND'
    if row['net_tkt_ct'] == 1:
        return 'ACTIVE'
    elif row['net_tkt_ct'] == -1:
        return 'REFUND'
    else:
        return 'NO-REFUND'

fhr_transactions['net_tkt_ct'] = fhr_transactions.apply(lambda row: get_net_tkt_ct(row), axis = 1)
eval_transactions['net_tkt_ct'] = eval_transactions.apply(lambda row: get_net_tkt_ct(row), axis = 1)
```

In [265]:
```
fhr_transactions.groupby(['trip_sta', 'prepay_in', 'net_tkt_ct'], dropna=False, as_index=False).size
()
```

Out[265]:

	trip_sta	prepay_in	net_tkt_ct	size
0	ACTIVE	UNKNOWN	ACTIVE	145452
1	CANCELLED	PAY LATER	ACTIVE	1310
2	CANCELLED	PAY NOW	ACTIVE	1005
3	CANCELLED	UNKNOWN	NO-REFUND	54367
4	INVOICED	PAY LATER	ACTIVE	251187
5	INVOICED	PAY LATER	NO-REFUND	45135
6	INVOICED	PAY LATER	REFUND	13
7	SUCCESS	PAY LATER	ACTIVE	2107
8	SUCCESS	PAY NOW	ACTIVE	4368

prog_id The id of the hotel

In [266]:
```
fhr_transactions.groupby(['prog_id'], dropna=False, as_index=False).size()
```

Out[266]:

	prog_id	size
0	FHC	90521
1	FHR	409768
2	THC	4597
3	NaN	58

card_type - Card type used for purchase

In [267]:
```
fhr_transactions.groupby(['card_type'], dropna=False, as_index=False).size()
```

Out[267]:

	card_type	size
0	AX	492702
1	PA	48
2	NaN	12194

cm_dma

In [268]:
```
len(fhr_transactions['cm_dma'].unique())
```

Out[268]:
```
209
```

cross_sell_type

In [269]:
```
fhr_transactions.groupby(['cross_sell_type'], dropna=False, as_index=False).size()
```

Out[269]:

	cross_sell_type	size
0	A+H	42176
1	A+H+C	3191
2	A+H+C+F	1067
3	A+H+C+F+O	5
4	A+H+C+F+O+T	10
5	A+H+C+F+S	1
6	A+H+C+F+S+T	4
7	A+H+C+F+T	75
8	A+H+C+O	102
9	A+H+C+O+S	10
10	A+H+C+O+S+T	9
11	A+H+C+O+T	13
12	A+H+C+S	25
13	A+H+C+S+T	3
14	A+H+C+T	658
15	A+H+F	14085
16	A+H+F+O	258
17	A+H+F+O+S	3
18	A+H+F+O+S+T	1
19	A+H+F+O+T	49
20	A+H+F+S	197
21	A+H+F+S+T	16
22	A+H+F+T	556
23	A+H+O	2893
24	A+H+O+S	21

	cross_sell_type	size
25	A+H+O+S+T	13
26	A+H+O+T	226
27	A+H+S	384
28	A+H+S+T	72
29	A+H+T	2871
30	H	424595
31	H+C	4364
32	H+C+F	2
33	H+C+F+O	1
34	H+C+F+O+T	1
35	H+C+F+T	28
36	H+C+O	23
37	H+C+O+S+T	1
38	H+C+O+T	22
39	H+C+S	9
40	H+C+T	674
41	H+F	2281
42	H+F+O	25
43	H+F+O+T	2
44	H+F+S	1
45	H+F+T	13
46	H+O	407
47	H+O+S	5
48	H+O+S+T	4
49	H+O+T	149
50	H+S	593

	cross_sell_type	size
51	H+S+T	52
52	H+T	2698

We have to separate these out to individual fields

In [270]:
```
cross_sell_type_fields = ['A', 'H', 'O', 'S', 'T']
for cross_sell_type_field in cross_sell_type_fields:
    cross_sell_type_field_name = "cross_sell_type_" + cross_sell_type_field
    fhr_transactions[cross_sell_type_field_name] = fhr_transactions['cross_sell_type'].apply(lambda x
: "YES" if cross_sell_type_field in x.split("+") else "NO")
    eval_transactions[cross_sell_type_field_name] = eval_transactions['cross_sell_type'].apply(lambda
x: "YES" if cross_sell_type_field in x.split("+") else "NO")
```

We can now drop this cross_see_type. #dropthis

In [271]:
```
drop_cols.append('cross_sell_type')
```

card_ctgy

32% of the values are null.

In [272]:
```
fhr_transactions['card_ctgy'].isna().sum()
```

Out[272]:
```
162885
```

There are other fields related to the card product. We can combine this with others.

In [273]:
```
fhr_transactions['card_ctgy'] = fhr_transactions.apply(lambda x: x['acct_type'] if pd.isnull(x['card_ctgy']) else x['card_ctgy'], axis = 1)
eval_transactions['card_ctgy'] = eval_transactions.apply(lambda x: x['acct_type'] if pd.isnull(x['card_ctgy']) else x['card_ctgy'], axis = 1)
```

pax_seq_no - Passenger Sequence Number

In [274]:
```
fhr_transactions['pax_seq_no'].isna().sum()
```

Out[274]: 0

Doesnt add much value. We can ignore. #dropthis

In [275]:
```
drop_cols.append('pax_seq_no')
```

srce_nm

In [276]:
```
fhr_transactions['srce_nm'].isna().sum()
```

Out[276]: 0

orig_state

In [277]:
```
fhr_transactions[fhr_transactions['orig_state'].isna()].groupby(['orig_ctry'], dropna=False, as_index
=False).size()
```

Out[277]:

	orig_ctry	size
0	ARGENTINA	640
1	AUSTRALIA	23072
2	AUSTRIA	1743
3	CANADA	20327
4	FINLAND	1007
5	FRANCE	6381
6	GERMANY	23525
7	HONG KONG	28106
8	INDIA	11670
9	ITALY	8275
10	JAPAN	34099
11	MEXICO	12847
12	NETHERLANDS	3505
13	NEW ZEALAND	275
14	NORWAY	54
15	SINGAPORE	7592
16	SPAIN	2476
17	SWEDEN	2702
18	TAIWAN	18450
19	THAILAND	4460
20	UNITED KINGDOM	46385
21	UNITED STATES	4061

In [278]:
```
fhr_transactions['orig_state'] = fhr_transactions.apply(lambda x: x['orig_state'] if pd.notnull(x['or
ig_state']) else x['orig_ctry'], axis = 1)
eval_transactions['orig_state'] = eval_transactions.apply(lambda x: x['orig_state'] if pd.notnull(x[
'orig_state']) else x['orig_ctry'], axis = 1)
```

orig_ctry

In [279]:
```
fhr_transactions['orig_ctry'].isna().sum()
```

Out[279]: 0

fare_waive_am - Code to waive fare

In [280]:
```
fhr_transactions['fare_waive_am'].isna().sum()
```

Out[280]: 499164

This field is not available for 99% of the rows. #dropthis

In [281]:
```
drop_cols.append('fare_waive_am')
```

pymt_form form of payment

In [282]:
```
fhr_transactions.groupby(['pymt_form'], dropna=False, as_index=False).size()
```

Out[282]:

	pymt_form	size
0	AR	101
1	CC	473810
2	CM	10889
3	NaN	20144

In [283]:
```
fhr_transactions.groupby(['pymt_form', 'pwp_book_flag'], dropna=False, as_index=False).size()
```

Out[283]:

	pymt_form	pwp_book_flag	size
0	AR	UNKNOWN	101
1	CC	UNKNOWN	473810
2	CM	UNKNOWN	10889
3	NaN	FULL	1186
4	NaN	NONE	5536
5	NaN	PARTIAL	1167
6	NaN	UNKNOWN	12255

1. if purchase was made with points - pwp_book_flag is FULL, then we can consider all those as OTHER pymt_form.
2. if purchase was made with points - pwp_book_flag is NONE, then we can consider all those as CC pymt_form.

In [284]:
```
fhr_transactions['pymt_form'] = fhr_transactions.apply(lambda x: "UNKNOWN" if pd.isnull(x['pymt_form'
]) else x['pymt_form'], axis = 1)
eval_transactions['pymt_form'] = eval_transactions.apply(lambda x: "UNKNOWN" if pd.isnull(x['pymt_for
m']) else x['pymt_form'], axis = 1)
```

local_curr_cd local currency code

In [285]:
```
fhr_transactions.groupby(['local_curr_cd'], dropna=False, as_index=False).size().head()
```

Out[285]:

	local_curr_cd	size
0	AED	7736
1	AUD	13810
2	AZN	133
3	BGN	2
4	BHD	206

Lets use country code as a proxy for currency code for missing values.

In [286]:
```
fhr_transactions['local_curr_cd'] = fhr_transactions.apply(lambda x: x['ctry_cd'] if pd.isna(x['local_curr_cd']) else x['local_curr_cd'], axis = 1)
eval_transactions['local_curr_cd'] = eval_transactions.apply(lambda x: x['ctry_cd'] if pd.isna(x['local_curr_cd']) else x['local_curr_cd'], axis = 1)
```

doc_sta - Status of the booking

In [287]:
```
fhr_transactions.groupby(['doc_sta'], dropna=False, as_index=False).size()
```

Out[287]:

	doc_sta	size
0	A	478873
1	N	866
2	Q	115
3	R	3565
4	S	264
5	V	460
6	X	552
7	Y	21
8	Z	84
9	NaN	20144

In [288]:
```
fhr_transactions.groupby(['doc_sta', 'trip_sta'], dropna=False, as_index=False).size()
```

Out[288]:

	doc_sta	trip_sta	size
0	A	ACTIVE	143705
1	A	CANCELLED	40081
2	A	INVOICED	295087
3	N	ACTIVE	57
4	N	CANCELLED	314
5	N	INVOICED	495
6	Q	ACTIVE	60
7	Q	CANCELLED	55
8	R	ACTIVE	526
9	R	CANCELLED	2992
10	R	INVOICED	47
11	S	ACTIVE	8
12	S	CANCELLED	4
13	S	INVOICED	252
14	V	ACTIVE	116
15	V	CANCELLED	280
16	V	INVOICED	64
17	X	ACTIVE	30
18	X	CANCELLED	132
19	X	INVOICED	390
20	Y	ACTIVE	3
21	Y	CANCELLED	18
22	Z	ACTIVE	28
23	Z	CANCELLED	56
24	NaN	ACTIVE	919

	doc_sta	trip_sta	size
25	NaN	CANCELLED	12750
26	NaN	SUCCESS	6475

In [289]:
```
fhr_transactions['doc_sta'] =  fhr_transactions['doc_sta'].apply(lambda x: "UNKNOWN" if pd.isnull(x)
else x)
eval_transactions['doc_sta'] =  eval_transactions['doc_sta'].apply(lambda x: "UNKNOWN" if pd.isnull(x
) else x)
```

Similar fields are - 'doc_sta', 'trip_sta', 'prepay_in', 'net_tkt_ct'

cmpnt_creat_dt - Component create date

In [290]:
```
fhr_transactions['cmpnt_creat_dt_month'] = fhr_transactions['cmpnt_creat_dt'].dt.month
eval_transactions['cmpnt_creat_dt_month'] = eval_transactions['cmpnt_creat_dt'].dt.month
```

We just need the month. We'll drop the year and the day #dropthis

In [291]:
```
drop_cols.append('cmpnt_creat_dt')
```

rgn_cd Region code

In [292]:
```
fhr_transactions.groupby(['rgn_cd'], dropna=False, as_index=False).size()
```

Out[292]:

	rgn_cd	size
0	EMEA	96053
1	JAPA	127724
2	LACC	33814
3	USA	247353

acct_nm Card name

In [293]:
```
fhr_transactions['acct_nm'].isna().sum()
```
Out[293]:
```
8900
```

In [294]:
```
fhr_transactions['acct_type'].isna().sum()
```
Out[294]:
```
35687
```

In [295]:
```
fhr_transactions['card_type'].isna().sum()
```
Out[295]:
```
12194
```

In [296]:
```
fhr_transactions['card_ctgy'].isna().sum()
```
Out[296]:
```
35567
```

In [297]:
```
fhr_transactions['book_card_type'].isna().sum()
```
Out[297]:
```
37393
```

Lets use card_ctgy to fill the missing rows in acct_nm

In [298]:
```
def get_acct_nm(row):
    if pd.notnull(row['acct_nm']):
        return row['acct_nm']

    if pd.notnull(row['card_ctgy']):
        return row['card_ctgy']

    if pd.notnull(row['acct_type']):
        return row['acct_type']

    if pd.notnull(row['book_card_type']):
        return row['book_card_type']

    return "UNKNOWN"


fhr_transactions['acct_nm'] = fhr_transactions.apply(lambda x: get_acct_nm(x), axis = 1)
eval_transactions['acct_nm'] = eval_transactions.apply(lambda x: get_acct_nm(x), axis = 1)
```

In [299]:
```
fhr_transactions['acct_nm'].isna().sum()
```

Out[299]: 0

htl_comm_in if there is a commision in the hotel

In [300]:
```
fhr_transactions.groupby(['htl_comm_in'], dropna=False, as_index=False).size()
```

Out[300]:

	htl_comm_in	size
0	N	944
1	Y	295391
2	NaN	208609

In [301]:
```
drop_cols.append('htl_comm_in')
```

htl_st_ad hotel street

In [302]:
```
fhr_transactions.groupby(['htl_st_ad'], dropna=False, as_index=False).size()
```

Out[302]:

	htl_st_ad	size
0	0 GEORGE STREET	109
1	0 OXFORD ROAD	5
2	0130 DAYBREAK RIDGE	234
3	0130 DAYBREAK RIDGE..	2
4	1 RUE SCRIBE	9
...	...	...
2130	ZANTE	2
2131	ZONE TOURISTIQUE CAP GAMMARTH	124
2132	ZONE TOURISTIQUE CAP GAMMARTH.	3
2133	ZORLU CENTER	47
2134	NaN	903

2135 rows × 2 columns

This field is not of any value and has high cardinality. #dropthis

In [303]:
```
drop_cols.append('htl_st_ad')
```

card_ctgy_grp Card Category

In [304]:
```
# fhr_transactions.groupby(['card_ctgy_grp'], dropna=False, as_index=False).size().head(5)
```

In [305]:
```
def get_card_ctgy_grp(row):
    categories = list()
    if pd.notnull(row['card_ctgy_grp']) & ('OTHER' in str(row['card_ctgy_grp'])):
        return 'OTHER'
    if pd.notnull(row['card_ctgy_grp']):
        return row['card_ctgy_grp']

    if 'BUSINESS' in str(row['acct_nm']):
        categories.append('BUSINESS')
    if 'PLATINUM' in str(row['acct_nm']):
        categories.append('PLATINUM')
    if 'CENTURION' in str(row['acct_nm']):
        categories.append('CENTURION')
    if len(categories) > 0:
        return ' '.join(categories)

    return 'OTHER'

fhr_transactions['card_ctgy_grp'] = fhr_transactions.apply(lambda x: get_card_ctgy_grp(x), axis = 1)
eval_transactions['card_ctgy_grp'] = eval_transactions.apply(lambda x: get_card_ctgy_grp(x), axis = 1
)
```

In [306]:
```
# fhr_transactions.groupby(['card_ctgy_grp'], dropna=False, as_index=False).size()
```

Lets use this field to fill missing values for card_ctgy also

In [307]:
```
# fhr_transactions[fhr_transactions['card_ctgy'].isna()].groupby(['card_ctgy_grp', 'cust_type', 'card
_ctgy'], dropna=False, as_index=False).size().sort_values(by = 'size')
```

In [308]:
```
fhr_transactions['card_ctgy'] = fhr_transactions.apply(lambda x: x['card_ctgy'] if pd.notna(x['card_c
tgy']) else x['card_ctgy_grp'], axis = 1)
eval_transactions['card_ctgy'] = eval_transactions.apply(lambda x: x['card_ctgy'] if pd.notna(x['card
_ctgy']) else x['card_ctgy_grp'], axis = 1)
```

In [309]:
```
fhr_transactions['card_ctgy'].isna().sum()
```

Out[309]: 0

Since we already have acct_nm, this field might be redundant. But lets retain because it has low cardinality.

htl_ctry_cd Country code of the hotel

In [310]:
```
fhr_transactions['htl_ctry_cd'].isna().sum()
```

Out[310]: 9253

In [311]:
```
fhr_transactions[fhr_transactions['htl_ctry_cd'].isna()].groupby(['ctry_cd'], dropna=False, as_index=False).size().head()
```

Out[311]:

	ctry_cd	size
0	ANGUILLA	2
1	ARGENTINA	2
2	ARUBA	2
3	AUSTRALIA	1
4	AUSTRIA	2

In [312]:
```
fhr_transactions['htl_ctry_cd'] = fhr_transactions.apply(lambda x: x['ctry_cd'] if pd.isnull(x['htl_ctry_cd']) else x['htl_ctry_cd'], axis = 1)
eval_transactions['htl_ctry_cd'] = eval_transactions.apply(lambda x: x['ctry_cd'] if pd.isnull(x['htl_ctry_cd']) else x['htl_ctry_cd'], axis = 1)
```

pseudo_city_cd Pseudo city code

In [313]:
```
fhr_transactions.groupby(['pseudo_city_cd'], dropna=False, as_index=False).size().head()
```

Out[313]:

	pseudo_city_cd	size
0	03TB	10338
1	0R0B	6472
2	0T70	14
3	0Z97	98
4	1XAH	4594

In [314]:
```
fhr_transactions.groupby(['srce_nm', 'pseudo_city_cd'], dropna=False, as_index=False).size().head()
```

Out[314]:

	srce_nm	pseudo_city_cd	size
0	CC	03TB	6737
1	CC	0R0B	3538
2	CC	0T70	13
3	CC	1XAH	3014
4	CC	1XCH	1428

agcy_nm Agency name

In [315]:
```
fhr_transactions.groupby(['agcy_nm'], dropna=False, as_index=False).size().head()
```

Out[315]:

	agcy_nm	size
0	[redacted]	52
1	[redacted]	2842
2	[redacted]	23
3	[redacted]	31
4	[redacted]	6699

In [316]:
```
fhr_transactions[fhr_transactions['chan_type'] == 'ONLINE'].groupby(['chan_type', 'agcy_nm'], dropna=
False, as_index=False).size().sort_values(by = 'size')
```

Out[316]:

	chan_type	agcy_nm	size
0	ONLINE	[redacted]	1
1	ONLINE	ONLINE	2
2	ONLINE	NaN	55528

In [317]:
```
fhr_transactions[fhr_transactions['chan_type'] == 'ONLINE'].groupby(['chan_type', 'agcy_nm'], dropna=
False, as_index=False).size().sort_values(by = 'size')
```

Out[317]:

	chan_type	agcy_nm	size
0	ONLINE	[redacted]	1
1	ONLINE	ONLINE	2
2	ONLINE	NaN	55528

This field is not populated for ONLINE channels.

Let's convert all the OFFLINE/NaNs to UNKNOWN.

In [318]:
```
fhr_transactions['agcy_nm'] = fhr_transactions.apply(lambda x: 'UNKNOWN' if (x['chan_type'] == 'OFFLINE') & pd.isnull(x['agcy_nm']) else x['agcy_nm'], axis = 1)
eval_transactions['agcy_nm'] = eval_transactions.apply(lambda x: 'UNKNOWN' if (x['chan_type'] == 'OFFLINE') & pd.isnull(x['agcy_nm']) else x['agcy_nm'], axis = 1)
```

In [319]:
```
fhr_transactions['agcy_nm'] = fhr_transactions.apply(lambda x: x['agcy_brnch_g6_cd'] if pd.isnull(x['agcy_nm']) else x['agcy_nm'], axis = 1)
eval_transactions['agcy_nm'] = eval_transactions.apply(lambda x: x['agcy_brnch_g6_cd'] if pd.isnull(x['agcy_nm']) else x['agcy_nm'], axis = 1)
```

In [320]:
```
fhr_transactions['agcy_nm'] = fhr_transactions.apply(lambda x: 'NOT-APPLICABLE' if x['chan_type'] == 'ONLINE' else x['agcy_nm'], axis = 1)
eval_transactions['agcy_nm'] = eval_transactions.apply(lambda x: 'NOT-APPLICABLE' if x['chan_type'] == 'ONLINE' else x['agcy_nm'], axis = 1)
```

In [321]:
```
fhr_transactions.groupby(['agcy_nm'], dropna=False, as_index=False).size().head()
```

Out[321]:

	agcy_nm	size
0	[redacted]	52
1	[redacted]	2842
2	[redacted]	23
3	[redacted]	31
4	[redacted]	6699

agcy_brnch_g6_cd Code of Agency branch

In [322]:
```
fhr_transactions.groupby(['agcy_brnch_g6_cd'], dropna=False, as_index=False).size().head()
```

Out[322]:

	agcy_brnch_g6_cd	size
0	0.0	2
1	2792.0	91
2	2797.0	2106
3	2803.0	66
4	3049.0	5

We can drop this field since it is not very accurate with agcy_nm and has significantly higher missing values. #dropthis

In [323]:
```
drop_cols.append('agcy_brnch_g6_cd')
```

acct_type Basic account type

In [324]:
```
import re
stop_words = ['THE', 'AMERICAN', 'EXPRESS', 'CARD', 'VISA', 'MASTERCARD', 'FROM', 'OTHER']

def get_acct_type(row):
    acct_type = row['acct_type']
    card_ctgy_grp = row['card_ctgy_grp']

    if pd.isnull(acct_type):
        if pd.notnull(card_ctgy_grp):
            acct_type = card_ctgy_grp
        else:
            acct_type = card_ctgy

    acct_type = re.sub(r'[^A-Za-z0-9 ]+', ' ', acct_type)
    acct_type_list = acct_type.split()
    result_set = set(acct_type_list) - set(stop_words)

    if len(result_set) == 0:
        return "NORMAL"
    else:
        return ' '.join(result_set)

fhr_transactions['acct_type'] = fhr_transactions.apply(lambda x: get_acct_type(x), axis = 1)
eval_transactions['acct_type'] = eval_transactions.apply(lambda x: get_acct_type(x), axis = 1)
```

In [325]:
```
# fhr_transactions.groupby(['acct_type'], dropna=False, as_index=False).size().head()
```

In [326]:
```
fhr_transactions[fhr_transactions['acct_type'].isna()].groupby(['acct_nm', 'card_ctgy', 'card_ctgy_gr
p'], dropna=False, as_index=False).size().sort_values(by = 'size')
```

Out[326]:

acct_nm	card_ctgy	card_ctgy_grp	size

cust_type Customer Type

In [327]:
```
# fhr_transactions.groupby(['cust_type'], dropna=False, as_index=False).size().head()
```

Lets identify the cust_type using card_ctgy_grp

In [328]:
```
fhr_transactions['cust_type'] = fhr_transactions.apply(lambda x: x['cust_type'] if pd.notnull(x['cust_type']) else x['card_ctgy_grp'], axis = 1)
eval_transactions['cust_type'] = eval_transactions.apply(lambda x: x['cust_type'] if pd.notnull(x['cust_type']) else x['card_ctgy_grp'], axis = 1)
```

In [329]:
```
# fhr_transactions.groupby(['cust_type'], dropna=False, as_index=False).size().head()
```

book_card_type Book card type

In [330]:
```
fhr_transactions['book_card_type'].isna().sum()
```

Out[330]:
```
37393
```

This field is redundant. #dropthis

In [331]:
```
drop_cols.append('book_card_type')
```

basic_supp_in Basic or Supp Indicator

In [332]:
```
fhr_transactions['basic_supp_in'].isna().sum()
```

Out[332]:
```
37393
```

This field is redundant. #dropthis

In [333]:
```
drop_cols.append('basic_supp_in')
```

vend_no Vendor No

In [334]:
```
fhr_transactions.groupby(['vend_cd', 'vend_no'], dropna=False, as_index=False).size().sort_values(by
= 'size')
```

Out[334]:

	vend_cd	vend_no	size
5195	50046	299900	1
4808	43643	900100.0	1
2667	26121	900100.0	1
2669	26177	54340.0	1
1132	146476	326100.0	1
...	...	...	...
4403	36422	55999.0	3137
5649	64086	299900.0	3279
4235	34070	299900.0	3651
2708	26358	55999.0	3928
5646	64086	55999.0	7660

6251 rows × 3 columns

dropthis. This field does not look accurate

In [335]:
```
drop_cols.append('vend_no')
```

htl_usd_rt - USD rate of the hotel

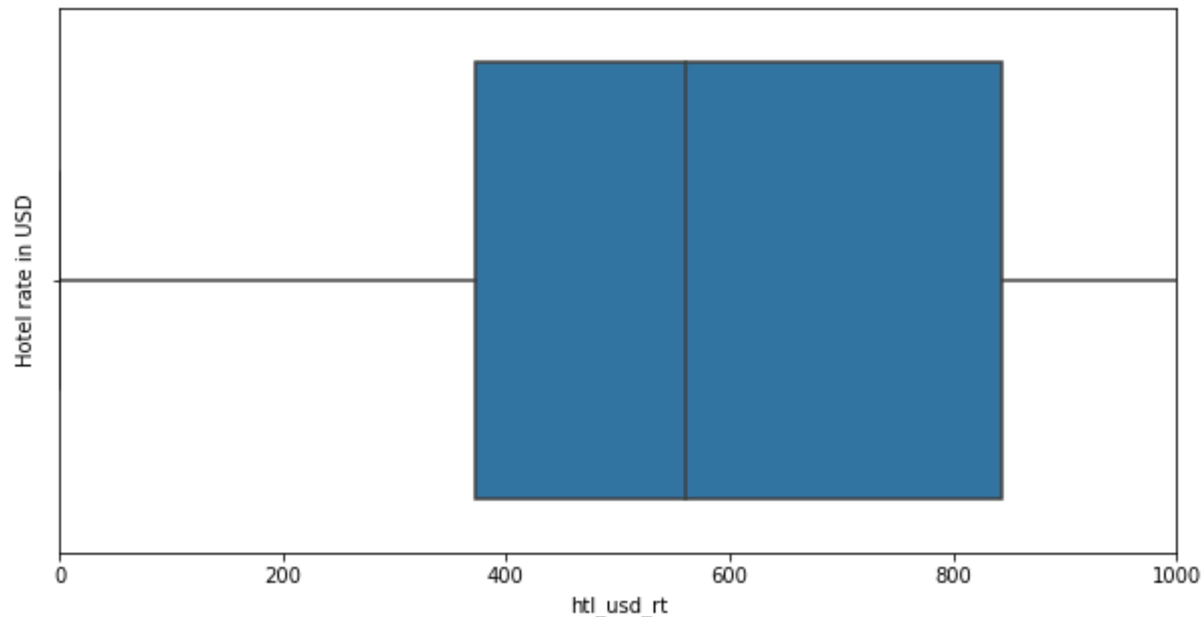
In [336]:
```
fhr_transactions['htl_usd_rt'] = fhr_transactions['htl_usd_rt'].apply(lambda x: np.NaN if pd.isna(x)
| (x <= 0) else x)
eval_transactions['htl_usd_rt'] = eval_transactions['htl_usd_rt'].apply(lambda x: np.NaN if pd.isna(x
) | (x <= 0) else x)
```

In [337]:
```
fhr_transactions['htl_usd_rt'].isna().sum()
```

Out[337]:
```
48
```

In [338]:
```
import numpy as np
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt

plt.figure(figsize=(10,5))
plt.xlim(0,1000)
plt.ylabel('Hotel rate in USD')
sns.boxplot(x=fhr_transactions['htl_usd_rt'])
plt.show()
```



Lets impute this using linear regression

In [339]:
```
fhr_transactions['htl_usd_rt'] = fhr_transactions['htl_usd_rt'].interpolate(method='linear', limit_direction='both', axis=0)
eval_transactions['htl_usd_rt'] = eval_transactions['htl_usd_rt'].interpolate(method='linear', limit_direction='both', axis=0)
```

In [340]:
```
#### trvl_ctry_dest_rgn Destination Region
```

In [341]:
```
fhr_transactions.groupby(['trvl_ctry_dest_rgn'], dropna=False, as_index=False).size().sort_values(by = 'size')
```

Out[341]:

	trvl_ctry_dest_rgn	size
9	NaN	204
3	CENTRAL AMERICA	907
0	AFRICA	2725
8	SOUTH AMERICA	5587
2	CARIBBEAN	8495
5	MIDDLE EAST	13303
7	OCEANIA	16107
1	ASIA	113110
4	EUROPE	133899
6	NORTH AMERICA	210607

In [342]:
```
fhr_transactions['trvl_ctry_dest_rgn'] = fhr_transactions.apply(lambda row: row['trvl_ctry_orig_rgn'] \
                                                                 if (row['dom_intl_in'] == 'D') & pd.isna(row['trvl_ctry_dest_rgn']) else row['trvl_ctry_dest_rgn'], axis = 1)
eval_transactions['trvl_ctry_dest_rgn'] = eval_transactions.apply(lambda row: row['trvl_ctry_orig_rgn'] \
                                                                 if (row['dom_intl_in'] == 'D') & pd.isna(row['trvl_ctry_dest_rgn']) else row['trvl_ctry_dest_rgn'], axis = 1)
```

In [343]:
```
fhr_transactions['trvl_ctry_dest_rgn'] = fhr_transactions.apply(lambda row: "UNKNOWN" if pd.isna(row[
'trvl_ctry_dest_rgn']) else row['trvl_ctry_dest_rgn'], axis = 1)
eval_transactions['trvl_ctry_dest_rgn'] = eval_transactions.apply(lambda row: "UNKNOWN" if pd.isna(ro
w['trvl_ctry_dest_rgn']) else row['trvl_ctry_dest_rgn'], axis = 1)
```

In [344]:
```
#### trvl_ctry_orig_rgn Origin Region
```

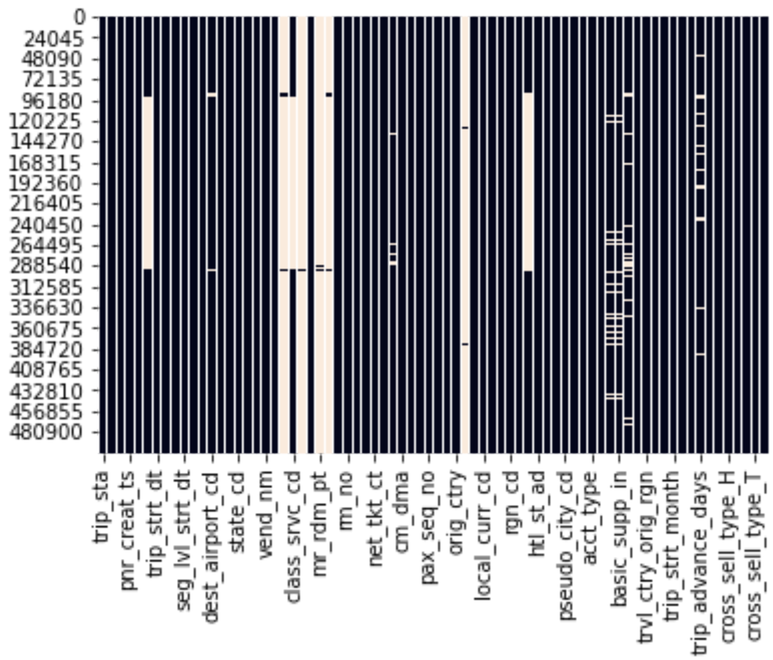
In [345]:
```
fhr_transactions.groupby(['trvl_ctry_orig_rgn'], dropna=False, as_index=False).size().sort_values(by
= 'size')
```

Out[345]:

	trvl_ctry_orig_rgn	size
4	SOUTH AMERICA	640
3	OCEANIA	23347
1	EUROPE	96053
0	ASIA	104377
2	NORTH AMERICA	280527

In [346]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(fhr_transactions.isnull(), cbar = False)
```

Out[346]: <AxesSubplot:>



Drop unwanted columns now

In [347]:
```
fhr_transactions.drop(columns=drop_cols, inplace=True)
eval_transactions.drop(columns=drop_cols, inplace=True)
```

In [348]:
```
fhr_transactions.shape
```

Out[348]: (504944, 48)

In [349]:
```
drop_cols
```

Out[349]:
```
['pnr_creat_ts',
 'trvl_ct',
 'trip_strt_dt',
 'trip_end_dt',
 'trip_diff_day',
 'seg_lvl_strt_dt',
 'seg_lvl_end_dt',
 'dest_airport_cd',
 'vend_nm',
 'class_cd',
 'class_srvc_cd',
 'pymt_pct',
 'mr_rdm_pt',
 'pkge_tax_am',
 'htl_rt',
 'trans_amt',
 'cross_sell_type',
 'pax_seq_no',
 'fare_waive_am',
 'cmpnt_creat_dt',
 'htl_comm_in',
 'htl_st_ad',
 'agcy_brnch_g6_cd',
 'book_card_type',
 'basic_supp_in',
 'vend_no']
```

In [350]:
```
fhr_transactions.describe(include='all')
```

Out[350]:

	trip_sta	mkt_cd	chan_type	person_id	trip_day_ct	dom_intl_in	city_nm	stat
count	504944	504944	504943	504944	504941.000000	504943	504944	50
unique	4	23	2	179218	NaN	2	563	
top	INVOICED	US	OFFLINE	8b112f7a6b00250a034e5967c266ea7379f9065c061ed8...	NaN	I	NEW YORK	CALIFO
freq	296335	247353	449412	494	NaN	257494	33528	3
mean	NaN	NaN	NaN	NaN	2.644139	NaN	NaN	
std	NaN	NaN	NaN	NaN	1.777291	NaN	NaN	
min	NaN	NaN	NaN	NaN	1.000000	NaN	NaN	
25%	NaN	NaN	NaN	NaN	1.000000	NaN	NaN	
50%	NaN	NaN	NaN	NaN	2.000000	NaN	NaN	
75%	NaN	NaN	NaN	NaN	3.000000	NaN	NaN	
max	NaN	NaN	NaN	NaN	65.000000	NaN	NaN	

In [351]:
```
corr = fhr_transactions.corr(method ='pearson')
corr.style.background_gradient(cmap='coolwarm').set_precision(2)
```

Out[351]:

	trip_day_ct	rm_no	htl_usd_rt	pnr_creat_ts_month	trip_strt_month	trip_end_month	seg_lvl_strt_month	trip_advar
trip_day_ct	1.00	0.01	0.20	-0.01	0.03	-0.01	0.03	
rm_no	0.01	1.00	-0.00	0.01	0.01	0.00	0.01	
htl_usd_rt	0.20	-0.00	1.00	-0.02	0.01	-0.01	0.01	
pnr_creat_ts_month	-0.01	0.01	-0.02	1.00	0.55	0.47	0.54	
trip_strt_month	0.03	0.01	0.01	0.55	1.00	0.86	0.97	
trip_end_month	-0.01	0.00	-0.01	0.47	0.86	1.00	0.89	
seg_lvl_strt_month	0.03	0.01	0.01	0.54	0.97	0.89	1.00	
trip_advance_days	0.19	0.01	0.10	-0.08	0.11	0.08	0.11	
seg_lvl_end_month	-0.01	0.00	-0.01	0.49	0.89	0.96	0.92	
cmpnt_creat_dt_month	0.02	0.00	0.00	0.83	0.66	0.58	0.66	

Based on the correlation matrix, lets drop the following 2 predictors

1. trip_strt_month
2. trip_end_month

In [352]:
```
drop_cols = ['trip_strt_month', 'trip_end_month']
fhr_transactions.drop(columns=drop_cols, inplace=True)
eval_transactions.drop(columns=drop_cols, inplace=True)
```

Also, lets convert the seg_lvl_strt_month and seg_lvl_end_month to categorical parameters

In [353]:
```
import calendar
fhr_transactions['seg_lvl_strt_month'] = fhr_transactions['seg_lvl_strt_month'].apply(lambda x: calen
dar.month_abbr[x])
eval_transactions['seg_lvl_end_month'] = eval_transactions['seg_lvl_end_month'].apply(lambda x: calen
dar.month_abbr[x])
```

Categorical Predictors Analysis

In [354]:
```
categorical_predictors = fhr_transactions.select_dtypes(include=['object']).columns.tolist()
```

In [355]:
```
fhr_transactions[categorical_predictors].describe(include='all').transpose()
```

Out[355]:

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

	count	unique	top	freq
card_ctgy_grp	504944	8	PLATINUM	234163
htl_ctry_cd	504944	166	USA	185665
pseudo_city_cd	504944	68	Z8B0	200556
agcy_nm	504944	126	NOT-APPLICABLE	55531
acct_type	504944	131	PLATINUM	229062
cust_type	504944	9	PLATINUM	249564
trvl_ctry_orig_rgn	504944	5	NORTH AMERICA	280527
trvl_ctry_dest_rgn	504944	10	NORTH AMERICA	210727
seg_lvl_strt_month	504944	12	Mar	50029
cross_sell_type_A	504944	2	NO	435950
cross_sell_type_H	504944	1	YES	504944
cross_sell_type_O	504944	2	NO	500691
cross_sell_type_S	504944	2	NO	503520
cross_sell_type_T	504944	2	NO	496724

Lets factorize the categorical predictors and find strongly correlated predictors

In [356]:
```
corr = fhr_transactions[categorical_predictors].apply(lambda x : pd.factorize(x)[0]).corr(method='pea
rson', min_periods=1)
corr.style.background_gradient(cmap='coolwarm').set_precision(2)
```

Out[356]:

	trip_sta	mkt_cd	chan_type	person_id	dom_intl_in	city_nm	state_cd	ctry_cd	vend_cd	vend_brand	pwp_book_fla
trip_sta	1.00	-0.04	0.04	-0.08	0.02	0.05	0.04	0.02	0.11	0.05	0.0
mkt_cd	-0.04	1.00	0.14	0.40	-0.31	0.12	0.13	0.23	0.17	-0.07	-0.1
chan_type	0.04	0.14	1.00	0.09	-0.06	0.14	0.15	0.08	0.15	0.24	0.3
person_id	-0.08	0.40	0.09	1.00	-0.15	0.06	0.07	0.11	0.07	-0.04	-0.0
dom_intl_in	0.02	-0.31	-0.06	-0.15	1.00	-0.09	-0.19	-0.31	-0.14	0.05	0.0
city_nm	0.05	0.12	0.14	0.06	-0.09	1.00	0.64	0.57	0.48	0.17	0.1
state_cd	0.04	0.13	0.15	0.07	-0.19	0.64	1.00	0.87	0.38	0.13	0.2
ctry_cd	0.02	0.23	0.08	0.11	-0.31	0.57	0.87	1.00	0.37	-0.02	-0.0
vend_cd	0.11	0.17	0.15	0.07	-0.14	0.48	0.38	0.37	1.00	0.23	0.1
vend_brand	0.05	-0.07	0.24	-0.04	0.05	0.17	0.13	-0.02	0.23	1.00	0.5
pwp_book_flag	0.06	-0.10	0.35	-0.04	0.09	0.18	0.20	-0.06	0.14	0.55	1.0
prepay_in	0.96	-0.05	0.01	-0.09	0.03	0.04	0.02	0.01	0.09	-0.00	-0.0
net_tkt_ct	-0.31	0.07	0.10	0.08	0.08	0.02	0.01	-0.01	-0.01	0.04	0.0
prog_id	-0.05	0.08	-0.05	0.04	-0.06	0.01	0.07	0.08	-0.02	-0.21	-0.0
card_type	-0.26	0.03	0.02	-0.01	0.02	0.00	0.00	0.01	-0.04	-0.03	0.0
cm_dma	0.05	-0.23	-0.05	-0.10	0.11	-0.04	-0.04	-0.10	-0.06	0.03	0.0
card_ctgy	-0.05	0.72	0.08	0.31	-0.23	0.10	0.11	0.20	0.14	-0.07	-0.0
srce_nm	0.96	-0.06	0.04	-0.10	0.04	0.06	0.04	0.01	0.10	0.05	0.0
orig_state	-0.05	0.81	0.21	0.39	-0.27	0.16	0.19	0.30	0.20	-0.07	-0.1
orig_ctry	-0.03	0.92	0.09	0.38	-0.28	0.13	0.16	0.27	0.18	-0.08	-0.1
pymt_form	0.26	-0.07	0.20	-0.01	0.05	0.11	0.13	-0.03	0.13	0.38	0.5
local_curr_cd	0.01	0.25	0.06	0.11	-0.15	0.38	0.61	0.69	0.28	-0.05	-0.0
doc_sta	0.29	-0.07	0.17	-0.02	0.01	0.10	0.12	-0.02	0.12	0.34	0.5
rgn_cd	-0.03	0.76	0.16	0.36	-0.18	0.19	0.23	0.35	0.23	-0.08	-0.1
acct_nm	-0.03	0.67	0.44	0.32	-0.22	0.18	0.21	0.29	0.21	0.00	-0.0

	trip_sta	mkt_cd	chan_type	person_id	dom_intl_in	city_nm	state_cd	ctry_cd	vend_cd	vend_brand	pwp_book_fla
card_ctgy_grp	0.03	-0.06	0.23	0.05	0.01	0.02	0.02	-0.00	0.01	0.02	0.0
htl_ctry_cd	0.06	0.14	0.27	0.07	-0.21	0.60	0.86	0.82	0.41	0.31	0.4
pseudo_city_cd	-0.04	1.00	0.19	0.41	-0.31	0.13	0.15	0.24	0.18	-0.05	-0.0
agcy_nm	-0.08	0.68	0.01	0.34	-0.19	0.14	0.17	0.24	0.17	-0.01	0.0
acct_type	0.01	0.24	0.07	0.10	-0.12	0.04	0.05	0.07	0.05	-0.01	-0.0
cust_type	0.03	-0.06	0.24	0.02	0.01	0.02	0.02	0.00	0.01	0.02	0.0
trvl_ctry_orig_rgn	-0.02	0.72	0.10	0.32	-0.11	0.19	0.23	0.34	0.23	-0.08	-0.1
trvl_ctry_dest_rgn	0.02	0.04	0.04	0.02	0.09	0.29	0.43	0.42	0.15	-0.01	-0.0
seg_lvl_strt_month	0.05	0.00	0.00	0.04	0.03	0.01	0.01	0.01	0.01	0.00	-0.0
cross_sell_type_A	-0.04	-0.03	0.13	0.01	0.16	-0.01	-0.05	-0.07	-0.02	0.03	0.0
cross_sell_type_H	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	na
cross_sell_type_O	0.01	0.07	-0.02	0.03	-0.04	0.02	0.03	0.04	0.02	-0.01	-0.0
cross_sell_type_S	0.00	-0.03	-0.02	0.00	-0.04	-0.01	0.00	0.01	-0.01	-0.01	-0.0
cross_sell_type_T	0.00	0.04	-0.04	-0.00	-0.05	0.00	0.00	0.01	-0.00	-0.01	-0.0

We observe the following.

1. srce_nm is strongly correlated with trip_sta
2. orig_ctry is strongly correlated with mkt_cd
3. pseudo_city_cd is strongly correlated with mkt_cd
4. rgn_cd is strongly correlated with orig_state
5. pseudo_city_cd is strongly correlated with orig_ctry
6. trvl_ctry_orig_rgn is strongly correlated with rgn_cd

Based on this, lets remove the 2 columns 'orig_ctry' and 'trvl_ctry_orig_rgn'

In [357]:
```
drop_cols = ['cross_sell_type_H']
#drop_cols = ['orig_ctry', 'trvl_ctry_orig_rgn', 'cross_sell_type_H']
fhr_transactions.drop(columns=drop_cols, inplace=True)
eval_transactions.drop(columns=drop_cols, inplace=True)
```

In [358]:
```
fhr_transactions.shape
```

Out[358]:
```
(504944, 45)
```

In [359]:
```
fhr_transactions.to_pickle('02fhr_fe1_output.pkl')
eval_transactions.to_pickle('02eval_fe1_output.pkl')
```

In [ ]: