In [64]:
```
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
import matplotlib.pyplot as plt

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

In [65]:
```
import logging
logger = logging.getLogger()
logger.setLevel(logging.CRITICAL)
```

Lets start with Exploratory Data Analysis

Import the dataset

In [66]:
```
fhr_transactions = pd.read_csv('../data/fhr_transactions.csv')
fhr_transactions.shape
```

Out[66]: (504944, 76)

Now, lets read the Evaluation Dataset as well.

In [67]:
```
eval_transactions = pd.read_csv('../../evaluation.csv').set_index('record_number')
```

Since the person_id is deemed sensitive, its masked using a one way hash algorithm

In [68]:
```
import hashlib
fhr_transactions['person_id'] = fhr_transactions['person_id'].astype(str).apply(lambda x: hashlib.sha
256(x.encode()).hexdigest())
eval_transactions['person_id'] = eval_transactions['person_id'].astype(str).apply(lambda x: hashlib.s
ha256(x.encode()).hexdigest())
```

Lets look at few rows from the dataframe

In [69]:
```
fhr_transactions.iloc[:, :15].head()
```

Out[69]:

	trip_sta	mkt_cd	chan_type	pnr_creat_ts	person_id	trvl_ct	trip_strt_dt	trip_end_dt	trip
0	INVOICED	US	OFFLINE	2018-04-05 00	f1705441121cc8e7addc35c65b9cd33994ce7356735fbb...	2.0	2018-04-25	2018-05-08	
1	INVOICED	US	OFFLINE	2018-04-05 00	f1705441121cc8e7addc35c65b9cd33994ce7356735fbb...	2.0	2018-04-25	2018-05-08	
2	INVOICED	US	OFFLINE	2018-04-05 00	f1705441121cc8e7addc35c65b9cd33994ce7356735fbb...	2.0	2018-04-25	2018-05-08	
3	INVOICED	US	OFFLINE	2018-04-05 00	aa2b23d8adff206808e83bca11030d2504c53c631a92d9...	2.0	2018-04-07	2018-04-08	
4	INVOICED	US	OFFLINE	2018-04-05 00	20795b1e182f5b74ee9e5c73019ec3d446d1c26bf2862f...	1.0	2018-04-05	2018-04-06	

Correcting values Globally

Lets convert all incorrect values to NaN so that we know more accurately

In [70]:
```
fhr_transactions.replace(['nan'], np.nan, inplace=True)
fhr_transactions.replace({r'[^\x00-\x7F]+':' '}, regex=True, inplace=True)

eval_transactions.replace(['nan'], np.nan, inplace=True)
eval_transactions.replace({r'[^\x00-\x7F]+':' '}, regex=True, inplace=True)
```

In [71]:
```
def trim_all_columns(df):
    """
    Trim whitespace from ends of each value across all series in dataframe
    """
    trim_strings = lambda x: x.strip() if isinstance(x, str) else x
    return df.applymap(trim_strings)

fhr_transactions = trim_all_columns(fhr_transactions)
eval_transactions = trim_all_columns(eval_transactions)
```

Lets find out how many NaN's are there in each column.

In [72]:
```
pd.set_option('display.max_rows', 500)
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),
                                    'unique': fhr_transactions.nunique()})
```

In [73]:
```
cols_df.sort_values('percent_missing')
```

Out[73]:

	column_name	number_missing	percent_missing	unique
trip_sta	trip_sta	0	0.000000	4
htl_rt	htl_rt	0	0.000000	17983
trans_amt	trans_amt	0	0.000000	29981
trvl_ctry_orig_rgn	trvl_ctry_orig_rgn	0	0.000000	5
cross_sell_type	cross_sell_type	0	0.000000	53
pax_seq_no	pax_seq_no	0	0.000000	82
srce_nm	srce_nm	0	0.000000	4
orig_ctry	orig_ctry	0	0.000000	22
vend_nm	vend_nm	0	0.000000	1441
vend_cd	vend_cd	0	0.000000	1438
cmpnt_creat_dt	cmpnt_creat_dt	0	0.000000	931
rgn_cd	rgn_cd	0	0.000000	4
vend_revn_grp	vend_revn_grp	0	0.000000	1
htl_night_ct	htl_night_ct	0	0.000000	40
pseudo_city_cd	pseudo_city_cd	0	0.000000	68
seg_lvl_strt_dt	seg_lvl_strt_dt	0	0.000000	1973
trip_diff_day	trip_diff_day	0	0.000000	1353
trip_end_dt	trip_end_dt	0	0.000000	2295
trip_strt_dt	trip_strt_dt	0	0.000000	2574
person_id	person_id	0	0.000000	179218
pnr_creat_ts	pnr_creat_ts	0	0.000000	6025
mkt_cd	mkt_cd	0	0.000000	23
rm_no	rm_no	0	0.000000	6
chan_type	chan_type	1	0.000198	2
trip_day_ct	trip_day_ct	3	0.000594	40

	column_name	number_missing	percent_missing	unique
seg_lvl_end_dt	seg_lvl_end_dt	3	0.000594	1986
htl_usd_rt	htl_usd_rt	4	0.000792	100810
trip_base_fare_usd	trip_base_fare_usd	4	0.000792	166719
trans_usd_am	trans_usd_am	4	0.000792	168854
tot_cmm_usd_am	tot_cmm_usd_am	4	0.000792	60780
local_curr_cd	local_curr_cd	25	0.004951	47
prog_id	prog_id	58	0.011486	3
city_nm	city_nm	196	0.038816	557
ctry_cd	ctry_cd	202	0.040004	119
trvl_ctry_dest_rgn	trvl_ctry_dest_rgn	204	0.040401	9
vend_brand	vend_brand	354	0.070107	121
unuse_tkt_flag	unuse_tkt_flag	901	0.178436	1
htl_st_ad	htl_st_ad	903	0.178832	2134
acct_nm	acct_nm	8900	1.762572	78
dest_airport_cd	dest_airport_cd	9251	1.832084	510
htl_ctry_cd	htl_ctry_cd	9253	1.832480	116
dom_intl_in	dom_intl_in	11724	2.321842	3
card_type	card_type	12194	2.414921	2
doc_sta	doc_sta	20144	3.989353	9
pymt_form	pymt_form	20144	3.989353	3
acct_type	acct_type	35687	7.067516	170
book_card_type	book_card_type	37393	7.405376	187
basic_supp_in	basic_supp_in	37393	7.405376	2
vend_no	vend_no	43421	8.599171	132
cust_type	cust_type	69964	13.855794	8
agcy_nm	agcy_nm	76884	15.226243	124

	column_name	number_missing	percent_missing	unique
card_ctgy	card_ctgy	162885	32.258033	185
card_ctgy_grp	card_ctgy_grp	194854	38.589230	10
class_srvc_cd	class_srvc_cd	198327	39.277029	1481
trvl_ct	trvl_ct	199819	39.572507	9
prepay_in	prepay_in	199819	39.572507	2
net_tkt_ct	net_tkt_ct	199819	39.572507	3
htl_comm_in	htl_comm_in	208609	41.313294	2
orig_state	orig_state	261652	51.818023	66
agcy_brnch_g6_cd	agcy_brnch_g6_cd	274090	54.281267	38
state_cd	state_cd	307030	60.804763	43
cm_dma	cm_dma	388298	76.899221	208
pkge_tax_am	pkge_tax_am	496154	98.259213	104
tax_conv_usd_am	tax_conv_usd_am	496154	98.259213	104
class_cd	class_cd	496154	98.259213	249
pwp_book_flag	pwp_book_flag	497055	98.437649	3
fare_waive_am	fare_waive_am	499164	98.855319	187
mr_rdm_pt	mr_rdm_pt	500264	99.073165	3337
pymt_pct	pymt_pct	502591	99.534008	101
refd_exch_in	refd_exch_in	502629	99.541533	1
flght_rdm	flght_rdm	504944	100.000000	0
non_refd_in	non_refd_in	504944	100.000000	0
misc_chrg_ord_in	misc_chrg_ord_in	504944	100.000000	0
expr_dt	expr_dt	504944	100.000000	0
net_remit_in	net_remit_in	504944	100.000000	0
snctn_ctry	snctn_ctry	504944	100.000000	0

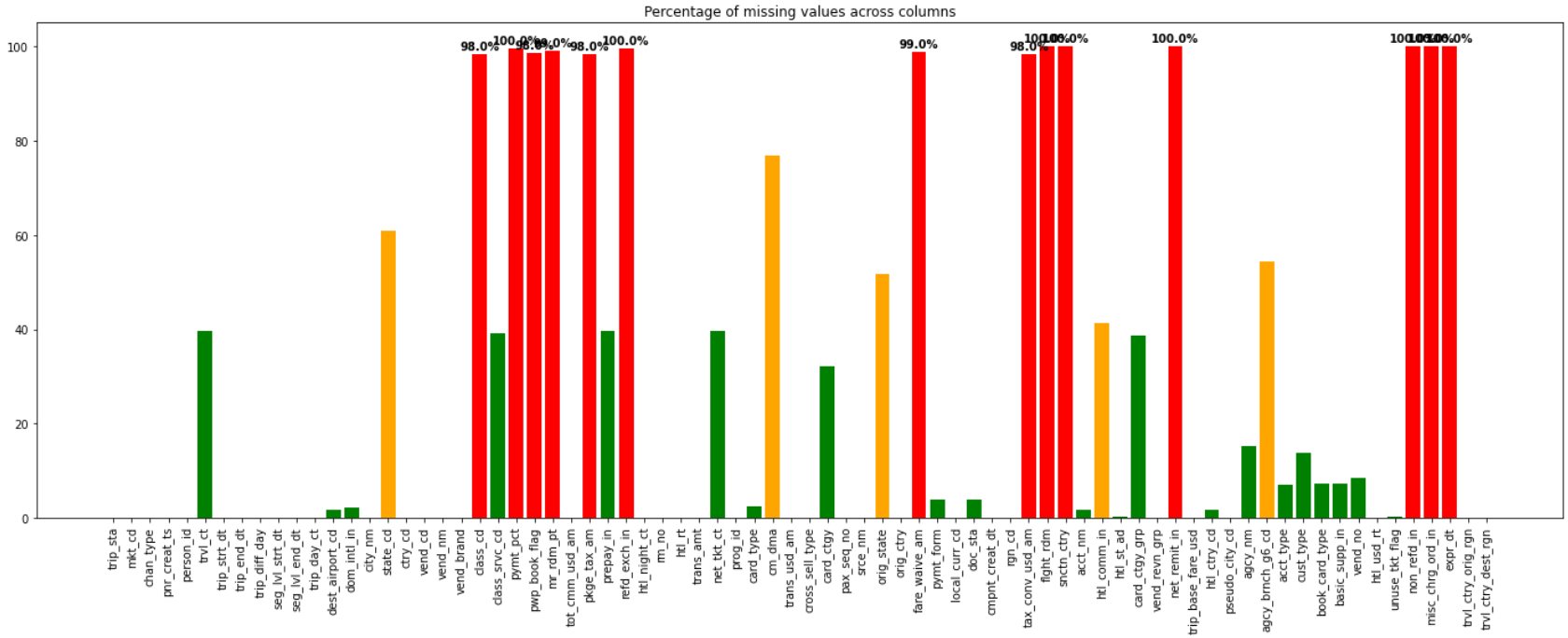
In [74]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(24,8))

colors_list = list()
for col_val in cols_df['percent_missing']:
    if col_val < 40:
        colors_list.append('green')
    elif col_val > 40 and col_val < 80:
        colors_list.append('orange')
    else:
        colors_list.append('red')

graph = plt.bar(cols_df.column_name, cols_df.percent_missing, color = colors_list)
plt.title('Percentage of missing values across columns')
plt.xticks(rotation='vertical')

i = 0
for p in graph:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    if (round(cols_df.percent_missing[i]) > 80):
        plt.text(x+width/2,
                 y+height*1.01,
                 str(round(cols_df.percent_missing[i]))+'%',
                 ha='center',
                 weight='bold')
    else:
        plt.text(x+width/2,
         y+height*1.01,
         "", ha='center', weight='bold')
    i+=1
plt.show()
```



In [75]:
```
# Add a bar chart
```

In [76]:
```
cols_df[cols_df['unique'] <= 1]['column_name'].tolist()
```

Out[76]:
```
['refd_exch_in',
 'flght_rdm',
 'snctn_ctry',
 'vend_revn_grp',
 'net_remit_in',
 'unuse_tkt_flag',
 'non_refd_in',
 'misc_chrg_ord_in',
 'expr_dt']
```

Lets remove all columns which do not add any value. i.e those which have 100% missing values or have the same value through-out

In [77]:
```
drop_cols = cols_df[cols_df['unique'] <= 1]['column_name'].tolist()
```

Lets look at the datatypes to see if they are in the right datatype.

In [78]:
```
fhr_transactions.sort_index(axis=1).info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 504944 entries, 0 to 504943
Data columns (total 76 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   acct_nm              496044 non-null  object 
 1   acct_type            469257 non-null  object 
 2   agcy_brnch_g6_cd     230854 non-null  float64
 3   agcy_nm              428060 non-null  object 
 4   basic_supp_in        467551 non-null  object 
 5   book_card_type       467551 non-null  object 
 6   card_ctgy            342059 non-null  object 
 7   card_ctgy_grp        310090 non-null  object 
 8   card_type            492750 non-null  object 
 9   chan_type            504943 non-null  object 
 10  city_nm              504748 non-null  object 
 11  class_cd             8790 non-null    object 
 12  class_srvc_cd        306617 non-null  object 
 13  cm_dma               116646 non-null  float64
 14  cmpnt_creat_dt       504944 non-null  object 
 15  cross_sell_type      504944 non-null  object 
 16  ctry_cd              504742 non-null  object 
 17  cust_type            434980 non-null  object 
 18  dest_airport_cd      495693 non-null  object 
 19  doc_sta              484800 non-null  object 
 20  dom_intl_in          493220 non-null  object 
 21  expr_dt              0 non-null       float64
 22  fare_waive_am        5780 non-null    object 
 23  flght_rdm            0 non-null       float64
 24  htl_comm_in          296335 non-null  object 
 25  htl_ctry_cd          495691 non-null  object 
 26  htl_night_ct         504944 non-null  int64  
 27  htl_rt               504944 non-null  float64
 28  htl_st_ad            504041 non-null  object 
 29  htl_usd_rt           504940 non-null  float64
 30  local_curr_cd        504919 non-null  object 
 31  misc_chrg_ord_in     0 non-null       float64
 32  mkt_cd               504944 non-null  object 
 33  mr_rdm_pt            4680 non-null    float64
 34  net_remit_in         0 non-null       float64
 35  net_tkt_ct           305125 non-null  float64
 36  non_refd_in          0 non-null       float64
 37  orig_ctry            504944 non-null  object 
```

```
 38  orig_state          243292 non-null  object
 39  pax_seq_no          504944 non-null  int64
 40  person_id           504944 non-null  object
 41  pkge_tax_am         8790 non-null    float64
 42  pnr_creat_ts        504944 non-null  object
 43  prepay_in           305125 non-null  object
 44  prog_id             504886 non-null  object
 45  pseudo_city_cd      504944 non-null  object
 46  pwp_book_flag       7889 non-null    object
 47  pymt_form           484800 non-null  object
 48  pymt_pct            2353 non-null    float64
 49  refd_exch_in        2315 non-null    object
 50  rgn_cd              504944 non-null  object
 51  rm_no               504944 non-null  int64
 52  seg_lvl_end_dt      504941 non-null  object
 53  seg_lvl_strt_dt     504944 non-null  object
 54  snctn_ctry          0 non-null       float64
 55  srce_nm             504944 non-null  object
 56  state_cd            197914 non-null  object
 57  tax_conv_usd_am     8790 non-null    float64
 58  tot_cmm_usd_am      504940 non-null  float64
 59  trans_amt           504944 non-null  float64
 60  trans_usd_am        504940 non-null  float64
 61  trip_base_fare_usd  504940 non-null  float64
 62  trip_day_ct         504941 non-null  float64
 63  trip_diff_day       504944 non-null  int64
 64  trip_end_dt         504944 non-null  object
 65  trip_sta            504944 non-null  object
 66  trip_strt_dt        504944 non-null  object
 67  trvl_ct             305125 non-null  float64
 68  trvl_ctry_dest_rgn  504740 non-null  object
 69  trvl_ctry_orig_rgn  504944 non-null  object
 70  unuse_tkt_flag      504043 non-null  object
 71  vend_brand          504590 non-null  object
 72  vend_cd             504944 non-null  int64
 73  vend_nm             504944 non-null  object
 74  vend_no             461523 non-null  object
 75  vend_revn_grp       504944 non-null  object
dtypes: float64(21), int64(5), object(50)
memory usage: 292.8+ MB
```

Lets convert columns to their correct datatype.

Lets convert Strings which are incorrectly identified as integers to Strings

In [79]:
```
str_cols = ['person_id', 'vend_cd', 'agcy_brnch_g6_cd', 'cm_dma']

for str_col in str_cols:
    fhr_transactions[str_col] = fhr_transactions[str_col].apply(str)
    eval_transactions[str_col] = eval_transactions[str_col].apply(str)
```

Lets convert the following date/time fields appropriately

In [80]:
```
datetime_cols = ['pnr_creat_ts', 'trip_strt_dt', 'trip_end_dt', 'seg_lvl_strt_dt', 'seg_lvl_end_dt',
'cmpnt_creat_dt']

for datetime_col in datetime_cols:
    fhr_transactions[datetime_col] = pd.DatetimeIndex(fhr_transactions[datetime_col])
    eval_transactions[datetime_col] = pd.DatetimeIndex(eval_transactions[datetime_col])
```

Lets describe all columns to find out the following.

1. The correct datatype.
2. Qualitative or Quantitative variable.
3. Cardinality of the categorical columns.

Find related columns by number of rows which have values.

In [81]:
```
cols_df[cols_df['number_missing'] > 0].sort_values(['number_missing', 'unique'])
```

Out[81]:

	column_name	number_missing	percent_missing	unique
chan_type	chan_type	1	0.000198	2
trip_day_ct	trip_day_ct	3	0.000594	40
seg_lvl_end_dt	seg_lvl_end_dt	3	0.000594	1986
tot_cmm_usd_am	tot_cmm_usd_am	4	0.000792	60780
htl_usd_rt	htl_usd_rt	4	0.000792	100810
trip_base_fare_usd	trip_base_fare_usd	4	0.000792	166719
trans_usd_am	trans_usd_am	4	0.000792	168854
local_curr_cd	local_curr_cd	25	0.004951	47
prog_id	prog_id	58	0.011486	3
city_nm	city_nm	196	0.038816	557
ctry_cd	ctry_cd	202	0.040004	119
trvl_ctry_dest_rgn	trvl_ctry_dest_rgn	204	0.040401	9
vend_brand	vend_brand	354	0.070107	121
unuse_tkt_flag	unuse_tkt_flag	901	0.178436	1
htl_st_ad	htl_st_ad	903	0.178832	2134
acct_nm	acct_nm	8900	1.762572	78
dest_airport_cd	dest_airport_cd	9251	1.832084	510
htl_ctry_cd	htl_ctry_cd	9253	1.832480	116
dom_intl_in	dom_intl_in	11724	2.321842	3
card_type	card_type	12194	2.414921	2
pymt_form	pymt_form	20144	3.989353	3
doc_sta	doc_sta	20144	3.989353	9
acct_type	acct_type	35687	7.067516	170
basic_supp_in	basic_supp_in	37393	7.405376	2
book_card_type	book_card_type	37393	7.405376	187

	column_name	number_missing	percent_missing	unique
vend_no	vend_no	43421	8.599171	132
cust_type	cust_type	69964	13.855794	8
agcy_nm	agcy_nm	76884	15.226243	124
card_ctgy	card_ctgy	162885	32.258033	185
card_ctgy_grp	card_ctgy_grp	194854	38.589230	10
class_srvc_cd	class_srvc_cd	198327	39.277029	1481
prepay_in	prepay_in	199819	39.572507	2
net_tkt_ct	net_tkt_ct	199819	39.572507	3
trvl_ct	trvl_ct	199819	39.572507	9
htl_comm_in	htl_comm_in	208609	41.313294	2
orig_state	orig_state	261652	51.818023	66
agcy_brnch_g6_cd	agcy_brnch_g6_cd	274090	54.281267	38
state_cd	state_cd	307030	60.804763	43
cm_dma	cm_dma	388298	76.899221	208
pkge_tax_am	pkge_tax_am	496154	98.259213	104
tax_conv_usd_am	tax_conv_usd_am	496154	98.259213	104
class_cd	class_cd	496154	98.259213	249
pwp_book_flag	pwp_book_flag	497055	98.437649	3
fare_waive_am	fare_waive_am	499164	98.855319	187
mr_rdm_pt	mr_rdm_pt	500264	99.073165	3337
pymt_pct	pymt_pct	502591	99.534008	101
refd_exch_in	refd_exch_in	502629	99.541533	1
flght_rdm	flght_rdm	504944	100.000000	0
snctn_ctry	snctn_ctry	504944	100.000000	0
net_remit_in	net_remit_in	504944	100.000000	0
non_refd_in	non_refd_in	504944	100.000000	0

	column_name	number_missing	percent_missing	unique
misc_chrg_ord_in	misc_chrg_ord_in	504944	100.000000	0
expr_dt	expr_dt	504944	100.000000	0

Now, lets also see which fields may be related to each other based on the number of values missing.

We see that the following fields are populated only for the same set of rows. 3 Predictors

1. tax_conv_usd_am
2. pkge_tax_am
3. class_cd

3 Predictors

1. trvl_ct
2. net_tkt_ct
3. prepay_in

2 Predictors

1. book_card_type
2. basic_supp_in

2 Predictors

1. doc_sta
2. pymt_form

2 Predictors

1. dest_airport_cd
2. htl_ctry_cd

Based on this, we see that tax_conv_usd_am and pkge_tax_am could have the same values. Lets test it out.

In [82]:
```
fhr_transactions['pkge_tax_am'].compare(fhr_transactions['tax_conv_usd_am'])
```

Out[82]:

	self	other

So lets drop one of them.

In [83]:
```
drop_cols.append('tax_conv_usd_am')
```

As part of Exploratory data analysis, lets find out if there are any duplicates in the dataset at all.

This is just by the whole dataset. During feature engineering, we may find out duplicates based on specific keys.

In [84]:
```
fhr_transactions.duplicated().sum()
```

Out[84]: 0

Lets drop all these cols from both of the dataframes.

In [85]:
```
fhr_transactions.drop(columns=drop_cols, inplace=True)
eval_transactions.drop(columns=drop_cols, inplace=True)
fhr_transactions.shape
```

Out[85]: (504944, 66)

In [86]:
```
drop_cols
```

Out[86]:
```
['refd_exch_in',
 'flght_rdm',
 'snctn_ctry',
 'vend_revn_grp',
 'net_remit_in',
 'unuse_tkt_flag',
 'non_refd_in',
 'misc_chrg_ord_in',
 'expr_dt',
 'tax_conv_usd_am']
```

Correlation

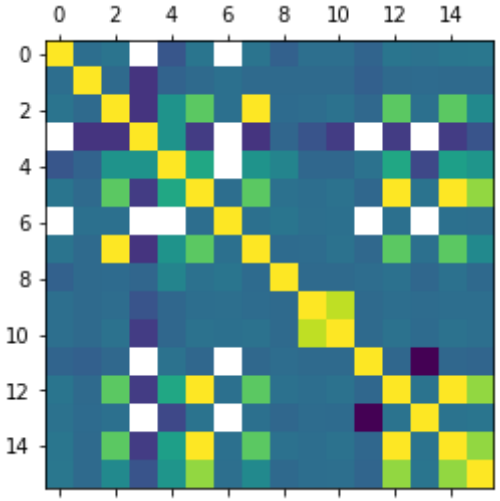
Lets find out of the continous predictors are correlated with others

In [87]:
```
fhr_transactions.corr(method ='pearson')
```

Out[87]:

	trvl_ct	trip_diff_day	trip_day_ct	pymt_pct	mr_rdm_pt	tot_cmm_usd_am	pkge_tax_am	htl_night_ct	rm_no	
trvl_ct	1.000000	0.022376	0.060117	NaN	-0.122441	0.062633	NaN	0.060117	-0.055310	(
trip_diff_day	0.022376	1.000000	0.001047	-0.300276	-0.045052	0.001404	0.036942	0.001041	-0.000495	-(
trip_day_ct	0.060117	0.001047	1.000000	-0.300276	0.254798	0.613501	0.036948	0.999883	0.006116	(
pymt_pct	NaN	-0.300276	-0.300276	1.000000	0.258703	-0.256083	NaN	-0.300276	-0.024583	-(
mr_rdm_pt	-0.122441	-0.045052	0.254798	0.258703	1.000000	0.381392	NaN	0.255945	0.161316	-(
tot_cmm_usd_am	0.062633	0.001404	0.613501	-0.256083	0.381392	1.000000	0.021713	0.613529	0.043613	(
pkge_tax_am	NaN	0.036942	0.036948	NaN	NaN	0.021713	1.000000	0.036948	0.066211	(
htl_night_ct	0.060117	0.001041	0.999883	-0.300276	0.255945	0.613529	0.036948	1.000000	0.006134	(
rm_no	-0.055310	-0.000495	0.006116	-0.024583	0.161316	0.043613	0.066211	0.006134	1.000000	-(
htl_rt	0.030752	-0.000394	0.014518	-0.133421	-0.016707	0.022100	0.027220	0.014495	-0.001673	1
trans_amt	0.027547	-0.000169	0.047452	-0.256083	-0.017004	0.044738	0.034913	0.047445	-0.000454	(
net_tkt_ct	-0.038798	-0.060324	-0.012353	NaN	0.046583	-0.015369	NaN	-0.012353	0.015823	(
trans_usd_am	0.062572	0.001400	0.613377	-0.256083	0.382626	0.999835	0.034913	0.613404	0.043753	(
pax_seq_no	0.048378	0.004391	0.037059	NaN	-0.199940	0.050694	NaN	0.037094	-0.018295	(
trip_base_fare_usd	0.068365	0.001429	0.612812	-0.256793	0.335768	0.999406	0.034276	0.612837	0.041604	(
htl_usd_rt	0.079268	0.001052	0.195961	-0.133421	0.268464	0.749657	0.027220	0.195926	-0.004743	(

In [88]:
```
plt.matshow(fhr_transactions.corr())
plt.show()
```



In [89]:
```
corr = fhr_transactions.corr(method ='pearson')
corr.style.background_gradient(cmap='coolwarm').set_precision(2)
```

Out[89]:

	trvl_ct	trip_diff_day	trip_day_ct	pymt_pct	mr_rdm_pt	tot_cmm_usd_am	pkge_tax_am	htl_night_ct	rm_no	htl_rt	t
trvl_ct	1.00	0.02	0.06	nan	-0.12	0.06	nan	0.06	-0.06	0.03	
trip_diff_day	0.02	1.00	0.00	-0.30	-0.05	0.00	0.04	0.00	-0.00	-0.00	
trip_day_ct	0.06	0.00	1.00	-0.30	0.25	0.61	0.04	1.00	0.01	0.01	
pymt_pct	nan	-0.30	-0.30	1.00	0.26	-0.26	nan	-0.30	-0.02	-0.13	
mr_rdm_pt	-0.12	-0.05	0.25	0.26	1.00	0.38	nan	0.26	0.16	-0.02	
tot_cmm_usd_am	0.06	0.00	0.61	-0.26	0.38	1.00	0.02	0.61	0.04	0.02	
pkge_tax_am	nan	0.04	0.04	nan	nan	0.02	1.00	0.04	0.07	0.03	
htl_night_ct	0.06	0.00	1.00	-0.30	0.26	0.61	0.04	1.00	0.01	0.01	
rm_no	-0.06	-0.00	0.01	-0.02	0.16	0.04	0.07	0.01	1.00	-0.00	
htl_rt	0.03	-0.00	0.01	-0.13	-0.02	0.02	0.03	0.01	-0.00	1.00	
trans_amt	0.03	-0.00	0.05	-0.26	-0.02	0.04	0.03	0.05	-0.00	0.86	
net_tkt_ct	-0.04	-0.06	-0.01	nan	0.05	-0.02	nan	-0.01	0.02	0.00	
trans_usd_am	0.06	0.00	0.61	-0.26	0.38	1.00	0.03	0.61	0.04	0.02	
pax_seq_no	0.05	0.00	0.04	nan	-0.20	0.05	nan	0.04	-0.02	0.01	
trip_base_fare_usd	0.07	0.00	0.61	-0.26	0.34	1.00	0.03	0.61	0.04	0.02	
htl_usd_rt	0.08	0.00	0.20	-0.13	0.27	0.75	0.03	0.20	-0.00	0.03	

Based on the above, we see that the following predictors are strongly correlated.

1. htl_night_ct is strongly correlated with trip_day_ct
2. trans_usd_am is strongly correlated with tot_cmm_usd_am
3. trip_base_fare_usd is strongly correlated with tot_cmm_usd_am
4. trip_base_fare_usd is strongly correlated with trans_usd_am

Lets see how which of these columns have valid values

In [90]:
```
fhr_transactions[['htl_night_ct', 'trip_day_ct', 'trans_usd_am', 'tot_cmm_usd_am', 'trip_base_fare_usd', 'trans_usd_am']].isna().sum()
```

Out[90]:
```
htl_night_ct          0
trip_day_ct           3
trans_usd_am          4
tot_cmm_usd_am        4
trip_base_fare_usd    4
trans_usd_am          4
dtype: int64
```

Lets drop the strongly correlated predictors

In [91]:
```
corr_cols = ['htl_night_ct', 'tot_cmm_usd_am', 'trip_base_fare_usd', 'trans_usd_am']
fhr_transactions.drop(columns=corr_cols, inplace=True)
eval_transactions.drop(columns=corr_cols, inplace=True)
```

We'll do the correlation for categorical predictors when we do feature analysis

Store it for Feature Engineering

Lets store this final dataframe in a pickle to use for Feature Engineering

In [92]:
```
fhr_transactions.to_pickle('01fhr_ede_output.pkl')
eval_transactions.to_pickle('01eval_ede_output.pkl')
```

In [93]:
```
fhr_transactions.shape
```

Out[93]:
```
(504944, 62)
```

In [ ]:

In [ ]:

In [ ]: