In [1]:

```
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt
import numpy as np
import datetime as dt

import missingno as msno
from textwrap import wrap
```

In [2]:

```
transaction_df = pd.read_excel('transaction.xlsx')
print(transaction_df.shape)
transaction_df.head().T
```

```
(20000, 13)
```

Out[2]:

	0	1	2	3	4
transaction_id	1	2	3	4	5
product_id	2	3	37	88	78
customer_id	2950	3120	402	3135	787
transaction_date	2017-02-25 00:00:00	2017-05-21 00:00:00	2017-10-16 00:00:00	2017-08-31 00:00:00	2017-10-01 00:00:00
online_order	0.0	1.0	0.0	0.0	1.0
order_status	Approved	Approved	Approved	Approved	Approved
brand	Solex	Trek Bicycles	OHM Cycles	Norco Bicycles	Giant Bicycles
product_line	Standard	Standard	Standard	Standard	Standard
product_class	medium	medium	low	medium	medium
product_size	medium	large	medium	medium	large
list_price	71.49	2091.47	1793.43	1198.46	1765.3
standard_cost	53.62	388.92	248.82	381.1	709.48
product_first_sold_date	41245.0	41701.0	36361.0	36145.0	42226.0

In [3]:

```
1
2 for col in ['order_status','brand','product_line','product_class','product_size'
3     print(col, transaction_df[col].unique())
4
```

```
order_status ['Approved' 'Cancelled']
brand ['Solex' 'Trek Bicycles' 'OHM Cycles' 'Norco Bicycles' 'Giant Bi
cycles'
 'WeareA2B' nan]
product_line ['Standard' 'Road' 'Mountain' 'Touring' nan]
product_class ['medium' 'low' 'high' nan]
product_size ['medium' 'large' 'small' nan]
```

In [4]:

```
df          = transaction_df.copy()
stats       = [df.isna().sum(), df.count(), df.nunique(),df.dtypes, df.min(), df.
stats_name = {0:'missing',
              1:'total',
              2:'unique',
              3:'dtypes',
              4:'minimum',
              5:'median',
              6:'maximum',
              7:'skewness'}

df_info = pd.DataFrame(stats).T.rename(columns=stats_name)
df_info
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: Fu
tureWarning: Dropping of nuisance columns in DataFrame reductions (wit
h 'numeric_only=None') is deprecated; in a future version this will ra
ise TypeError.  Select only valid columns before calling the reductio
n.

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: Fu
tureWarning: DataFrame.mean and DataFrame.median with numeric_only=Non
e will include datetime64 and datetime64tz columns in a future versio
n.
```

Out[4]:

	missing	total	unique	dtypes	minimum	median	maximum	sk
transaction_id	0	20000	20000	int64	1	10000.5	20000	
product_id	0	20000	101	int64	0	44.0	100	0
customer_id	0	20000	3494	int64	1	1736.0	5034	
transaction_date	0	20000	364	datetime64[ns]	2017-01-01 00:00:00	NaN	2017-12-30 00:00:00	
online_order	360	19640	2	float64	0.0	1.0	1.0	-0
order_status	0	20000	2	object	Approved	NaN	Cancelled	
brand	197	19803	6	object	NaN	NaN	NaN	
product_line	197	19803	4	object	NaN	NaN	NaN	
product_class	197	19803	3	object	NaN	NaN	NaN	
product_size	197	19803	3	object	NaN	NaN	NaN	
list_price	0	20000	296	float64	12.01	1163.89	2091.47	-0
standard_cost	197	19803	103	float64	7.21	507.58	1759.85	0
product_first_sold_date	197	19803	100	float64	33259.0	38216.0	42710.0	-0

In [5]:

```
import pandas as pd
import seaborn as sns
from tqdm.notebook import tqdm
import warnings
warnings.simplefilter('ignore')
tqdm.pandas()
```

In [6]:

```

def prepare_data(df, select_columns):
    df              = df[select_columns]
    df['margin']    = df['list_price'] - df['standard_cost']
    return df

df = pd.read_excel('transaction.xlsx')
select_columns = ['transaction_date', 'customer_id','list_price','standard_cost'
df = prepare_data(df, select_columns)
df.head(2)

```

Out[6]:

	transaction_date	customer_id	list_price	standard_cost	margin
0	2017-02-25	2950	71.49	53.62	17.87
1	2017-05-21	3120	2091.47	388.92	1702.55

In [7]:

```

def get_cohort_timestamp(df, col_name, new_col_name):
    transaction_year  = df[col_name].dt.year
    transaction_month = df[col_name].dt.month
    df[new_col_name]  = [dt.datetime(i,j,1) for i,j in zip(transaction_year, tra

    return df

df = get_cohort_timestamp(df, 'transaction_date', 'cohort_timestamp')
df.head()

```

Out[7]:

	transaction_date	customer_id	list_price	standard_cost	margin	cohort_timestamp
0	2017-02-25	2950	71.49	53.62	17.87	2017-02-01
1	2017-05-21	3120	2091.47	388.92	1702.55	2017-05-01
2	2017-10-16	402	1793.43	248.82	1544.61	2017-10-01
3	2017-08-31	3135	1198.46	381.10	817.36	2017-08-01
4	2017-10-01	787	1765.30	709.48	1055.82	2017-10-01

In [8]:

```
def add_cohort_index(df, transaction_column, cohort_column, focus_column):
    gpy_info           = df.groupby(by = focus_column).min()[cohort_column].rese
    df                 = df.drop(columns = cohort_column).merge(gpy_info, on=foc
    year_diff          = df[transaction_column].dt.year  - df[cohort_column].dt.
    month_diff         = df[transaction_column].dt.month - df[cohort_column].dt.
    df['cohort_index'] = [12*i + j + 1 for i, j in zip(year_diff, month_diff)]
    return df


df = add_cohort_index(df, 'transaction_date', 'cohort_timestamp', 'customer_id')
df.head()

```

Out[8]:

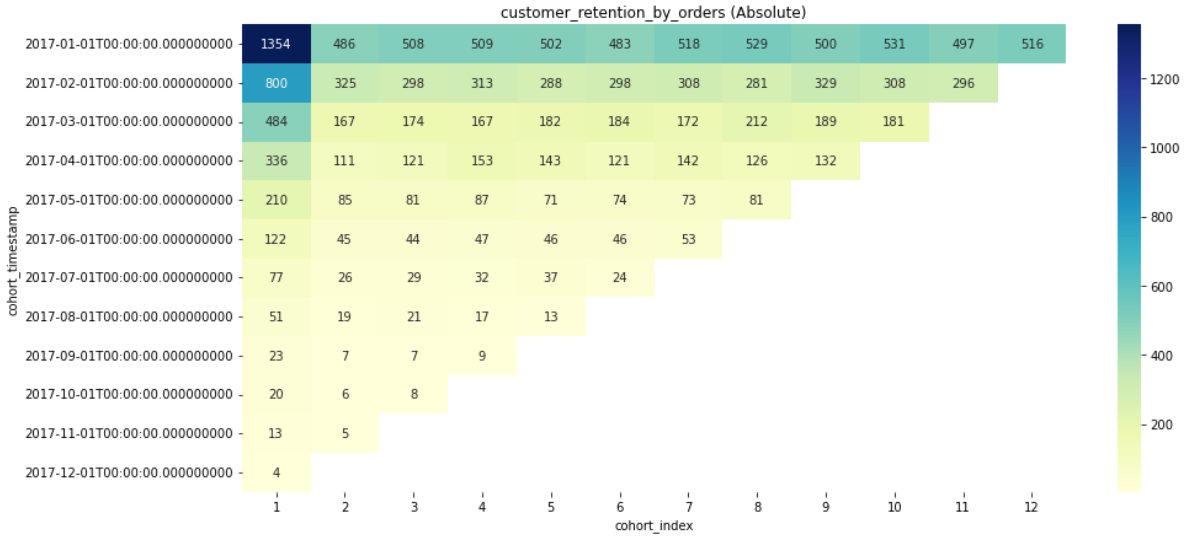
	transaction_date	customer_id	list_price	standard_cost	margin	cohort_timestamp	cohort_in
0	2017-02-25	2950	71.49	53.62	17.87	2017-02-01	
1	2017-10-16	2950	1403.50	954.82	448.68	2017-02-01	
2	2017-04-26	2950	478.16	298.72	179.44	2017-02-01	
3	2017-05-21	3120	2091.47	388.92	1702.55	2017-01-01	
4	2017-10-05	3120	1129.13	677.48	451.65	2017-01-01	

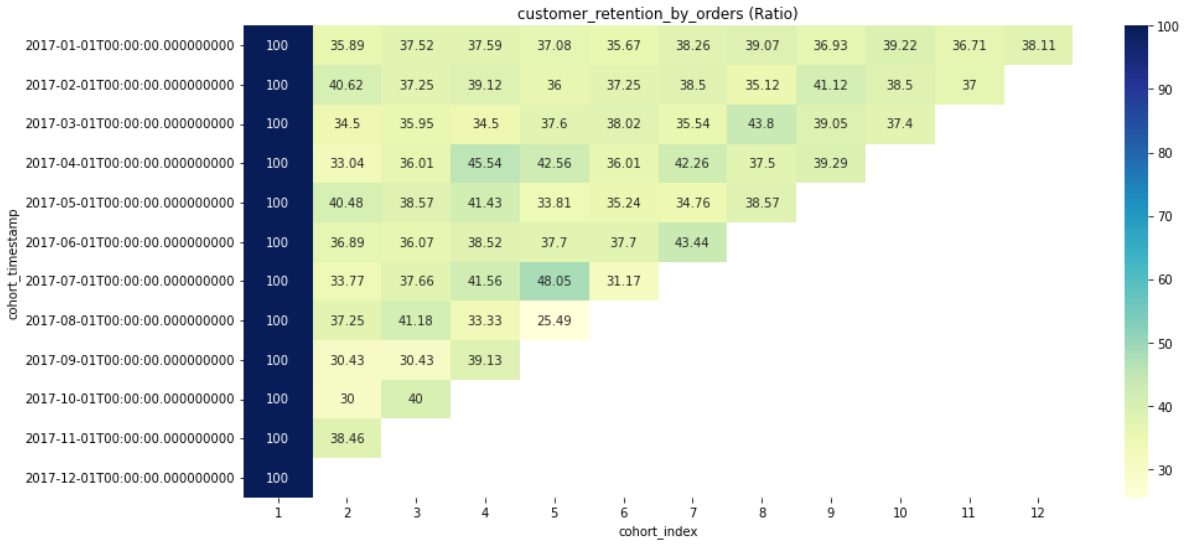
In [9]:

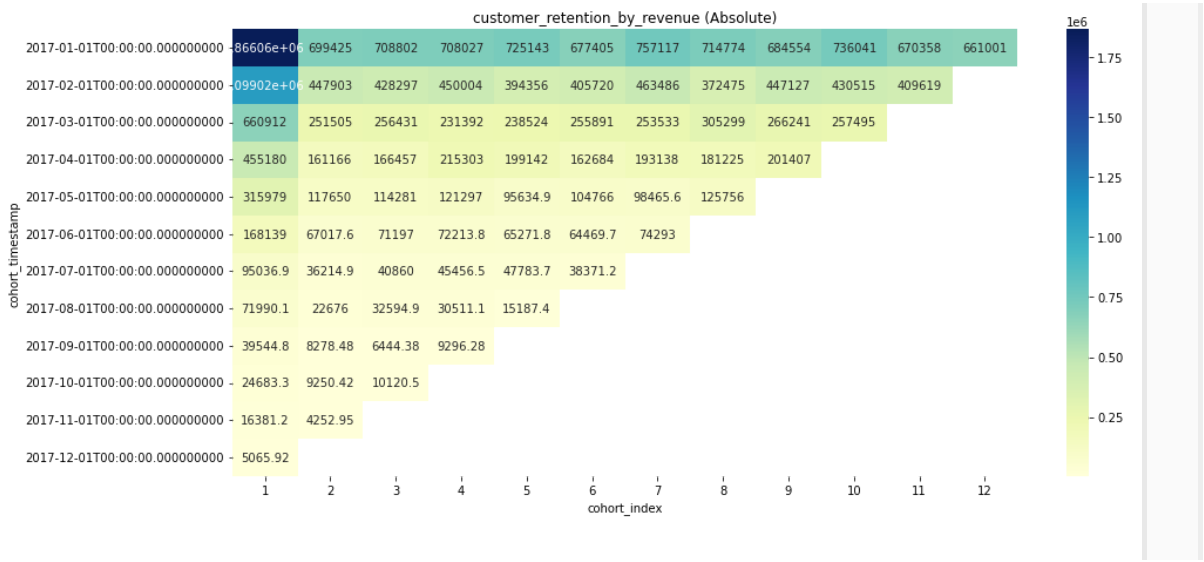
```
def plot_heatmap(df, title=''):
    plt.figure(figsize=(15,7))
    plt.title(title)
    sns.heatmap(df, cmap='YlGnBu', annot=True, fmt='g')
    plt.show()

```

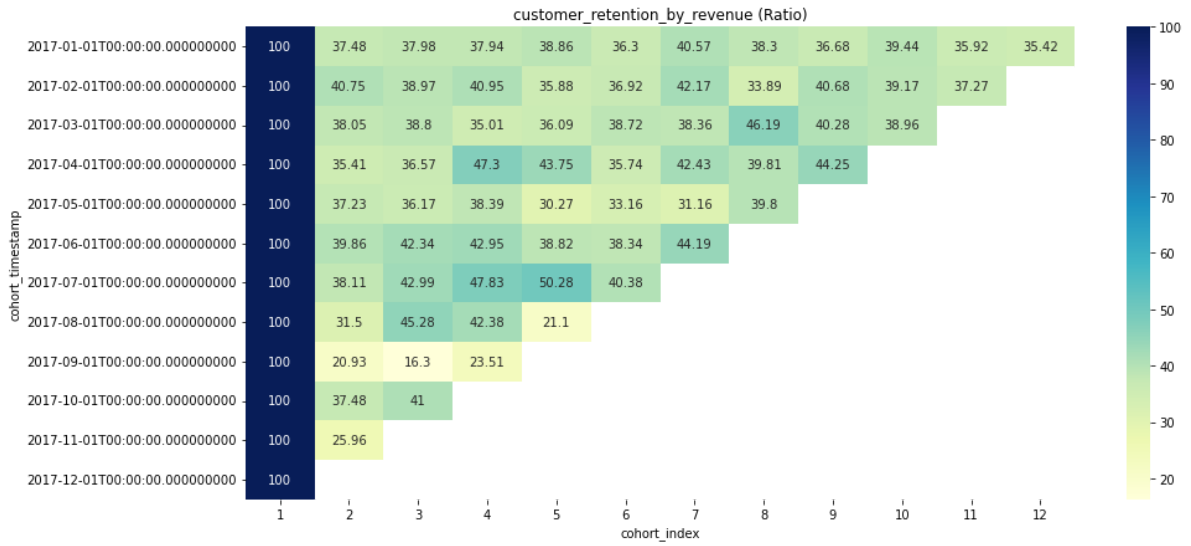
In [10]:

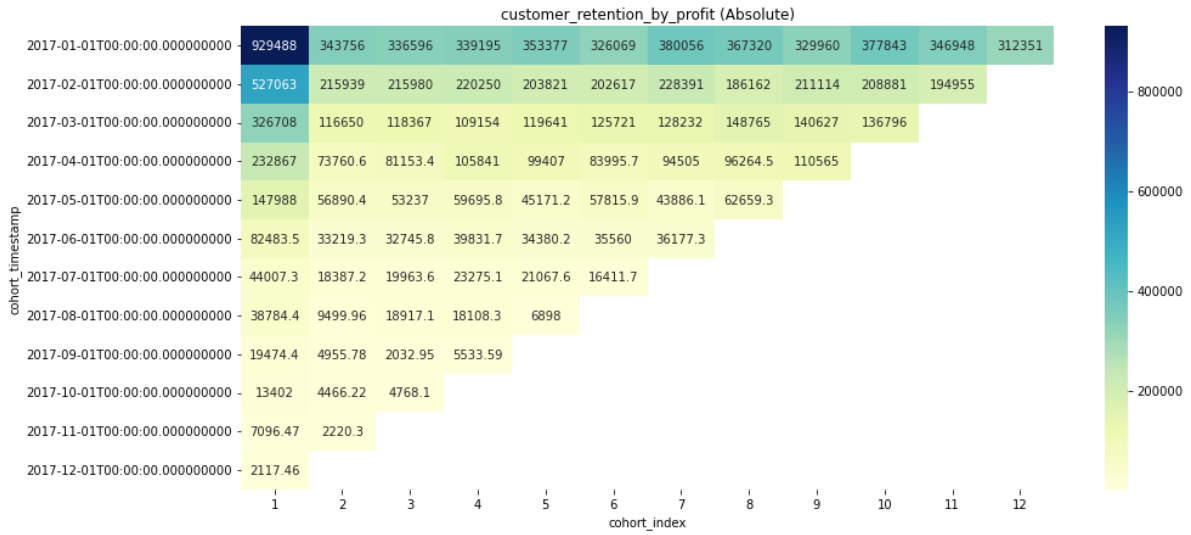
```
def get_chort_chart(df, row, column, value, function):

    if function == 'customer_retention_by_orders':
        table  = df.groupby(by = [row, column]).nunique()[value]

    elif function in ['customer_retention_by_revenue', 'customer_retention_by_pr
        table  = df.groupby(by = [row, column]).sum()[value]

    data     = pd.DataFrame(table).reset_index()
    absolute = data.pivot(index = row, columns = column, values=value)
    ratio    = round(absolute.divide(absolute.iloc[:,0], axis=0)*100,2)

    plot_heatmap(absolute, function+' (Absolute)')
    plot_heatmap(ratio, function+' (Ratio)')


get_chort_chart(df, 'cohort_timestamp','cohort_index', 'customer_id', 'customer_
get_chort_chart(df, 'cohort_timestamp','cohort_index', 'list_price' , 'customer_
get_chort_chart(df, 'cohort_timestamp','cohort_index', 'margin'     , 'customer_
```

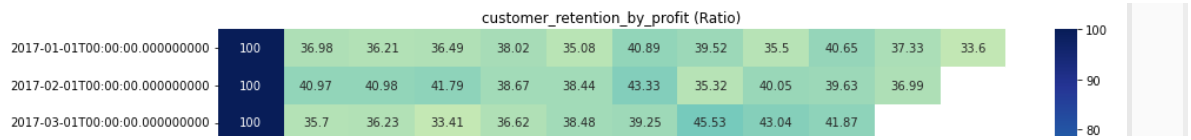





customer_retention_by_revenue (Absolute)
cohort_timestamp
2017-01-01T00:00:00.000000000 86606e+06 699425 708802 708027 725143 677405 757117 714774 684554 736041 670358 661001
2017-02-01T00:00:00.000000000 09902e+06 447903 428297 450004 394356 405720 463486 372475 447127 430515 409619
2017-03-01T00:00:00.000000000 660912 251505 256431 231392 238524 255891 253533 305299 266241 257495
2017-04-01T00:00:00.000000000 455180 161166 166457 215303 199142 162684 193138 181225 201407
2017-05-01T00:00:00.000000000 315979 117650 114281 121297 95634.9 104766 98465.6 125756
2017-06-01T00:00:00.000000000 168139 67017.6 71197 72213.8 65271.8 64469.7 74293
2017-07-01T00:00:00.000000000 95036.9 36214.9 40860 45456.5 47783.7 38371.2
2017-08-01T00:00:00.000000000 71990.1 22676 32594.9 30511.1 15187.4
2017-09-01T00:00:00.000000000 39544.8 8278.48 6444.38 9296.28
2017-10-01T00:00:00.000000000 24683.3 9250.42 10120.5
2017-11-01T00:00:00.000000000 16381.2 4252.95
2017-12-01T00:00:00.000000000 5065.92
1 2 3 4 5 6 7 8 9 10 11 12
cohort_index
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25


customer_retention_by_revenue (Ratio)
cohort_timestamp
2017-01-01T00:00:00.000000000 100 37.48 37.98 37.94 38.86 36.3 40.57 38.3 36.68 39.44 35.92 35.42
2017-02-01T00:00:00.000000000 100 40.75 38.97 40.95 35.88 36.92 42.17 33.89 40.68 39.17 37.27
2017-03-01T00:00:00.000000000 100 38.05 38.8 35.01 36.09 38.72 38.36 46.19 40.28 38.96
2017-04-01T00:00:00.000000000 100 35.41 36.57 47.3 43.75 35.74 42.43 39.81 44.25
2017-05-01T00:00:00.000000000 100 37.23 36.17 38.39 30.27 33.16 31.16 39.8
2017-06-01T00:00:00.000000000 100 39.86 42.34 42.95 38.82 38.34 44.19
2017-07-01T00:00:00.000000000 100 38.11 42.99 47.83 50.28 40.38
2017-08-01T00:00:00.000000000 100 31.5 45.28 42.38 21.1
2017-09-01T00:00:00.000000000 100 20.93 16.3 23.51
2017-10-01T00:00:00.000000000 100 37.48 41
2017-11-01T00:00:00.000000000 100 25.96
2017-12-01T00:00:00.000000000 100
1 2 3 4 5 6 7 8 9 10 11 12
cohort_index
100
90
80
70
60
50
40
30
20


customer_retention_by_profit (Absolute)
cohort_timestamp
2017-01-01T00:00:00.000000000 929488 343756 336596 339195 353377 326069 380056 367320 329960 377843 346948 312351
2017-02-01T00:00:00.000000000 527063 215939 215980 220250 203821 202617 228391 186162 211114 208881 194955
2017-03-01T00:00:00.000000000 326708 116650 118367 109154 119641 125721 128232 148765 140627 136796
2017-04-01T00:00:00.000000000 232867 73760.6 81153.4 105841 99407 83995.7 94505 96264.5 110565
2017-05-01T00:00:00.000000000 147988 56890.4 53237 59695.8 45171.2 57815.9 43886.1 62659.3
2017-06-01T00:00:00.000000000 82483.5 33219.3 32745.8 39831.7 34380.2 35560 36177.3
2017-07-01T00:00:00.000000000 44007.3 18387.2 19963.6 23275.1 21067.6 16411.7
2017-08-01T00:00:00.000000000 38784.4 9499.96 18917.1 18108.3 6898
2017-09-01T00:00:00.000000000 19474.4 4955.78 2032.95 5533.59
2017-10-01T00:00:00.000000000 13402 4466.22 4768.1
2017-11-01T00:00:00.000000000 7096.47 2220.3
2017-12-01T00:00:00.000000000 2117.46
1 2 3 4 5 6 7 8 9 10 11 12
cohort_index
800000
600000
400000
200000


customer_retention_by_profit (Ratio)
2017-01-01T00:00:00.000000000
100 36.98 36.21 36.49 38.02 35.08 40.89 39.52 35.5 40.65 37.33 33.6
2017-02-01T00:00:00.000000000
100 40.97 40.98 41.79 38.67 38.44 43.33 35.32 40.05 39.63 36.99
2017-03-01T00:00:00.000000000
100 35.7 36.23 33.41 36.62 38.48 39.25 45.53 43.04 41.87
100
90
80

In [ ]:

```
1
```

In [ ]:

```
1
```