```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True
```

Load required libraries and datasets

```
! cp drive/My\ Drive/QVI_data.csv .
```

```
import pandas as pd
```

```
import plotly.express as px
```

```
import numpy as np
```

```
df=pd.read_csv('QVI_data.csv')
```

```
df.shape
```

```
(264834, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264834 entries, 0 to 264833
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   LYLTY_CARD_NBR    264834 non-null  int64  
 1   DATE              264834 non-null  object 
 2   STORE_NBR         264834 non-null  int64  
 3   TXN_ID            264834 non-null  int64  
 4   PROD_NBR          264834 non-null  int64  
 5   PROD_NAME         264834 non-null  object 
 6   PROD_QTY          264834 non-null  int64  
 7   TOT_SALES         264834 non-null  float64
 8   PACK_SIZE         264834 non-null  int64  
 9   BRAND             264834 non-null  object 
 10  LIFESTAGE         264834 non-null  object 
 11  PREMIUM_CUSTOMER  264834 non-null  object 
dtypes: float64(1), int64(6), object(5)
memory usage: 24.2+ MB
```

```
df.describe(include= 'all')
```

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE
count	2.648340e+05	264834	264834.000000	2.648340e+05	264834.000000	264834	264834.000000	264834.000000	264834.000000
unique	NaN	364	NaN	NaN	NaN	114	NaN	NaN	NaN
top	NaN	2018–12–24	NaN	NaN	NaN	Kettle Mozzarella Basil & Pesto 175g	NaN	NaN	NaN
freq	NaN	939	NaN	NaN	NaN	3304	NaN	NaN	NaN
mean	1.355488e+05	NaN	135.079423	1.351576e+05	56.583554	NaN	1.905813	7.299346	182.425512
std	8.057990e+04	NaN	76.784063	7.813292e+04	32.826444	NaN	0.343436	2.527241	64.325148
min	1.000000e+03	NaN	1.000000	1.000000e+00	1.000000	NaN	1.000000	1.500000	70.000000
25%	7.002100e+04	NaN	70.000000	6.760050e+04	28.000000	NaN	2.000000	5.400000	150.000000
50%	1.303570e+05	NaN	130.000000	1.351365e+05	56.000000	NaN	2.000000	7.400000	170.000000
75%	2.030940e+05	NaN	203.000000	2.026998e+05	85.000000	NaN	2.000000	9.200000	175.000000
max	2.373711e+06	NaN	272.000000	2.415841e+06	114.000000	NaN	5.000000	29.500000	380.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264834 entries, 0 to 264833
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   LYLTY_CARD_NBR    264834 non-null  int64
 1   DATE              264834 non-null  object
 2   STORE_NBR         264834 non-null  int64
 3   TXN_ID            264834 non-null  int64
 4   PROD_NBR          264834 non-null  int64
 5   PROD_NAME         264834 non-null  object
 6   PROD_QTY          264834 non-null  int64
 7   TOT_SALES         264834 non-null  float64
 8   PACK_SIZE         264834 non-null  int64
 9   BRAND             264834 non-null  object
```

Trial store 77

Select control store

Add Month column

```
import datetime


df['year'] = pd.DatetimeIndex(df['DATE']).year
df['month']=pd.DatetimeIndex(df['DATE']).month
df['year_month']=pd.to_datetime(df['DATE']).dt.floor('d') - pd.offsets.MonthBegin(1)
df
```

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	L
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NATURAL	SINGLES/
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	RRD	SINGLES/
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	GRNWVES	YOUNG
3	1003	2019-03-08	1	4	106	Natural ChipCo Hony Soy Chckn175g	1	3.0	175	NATURAL	YOUNG
4	1004	2018-11-02	1	5	96	WW Original Stacked Chips 160g	1	1.9	160	WOOLWORTHS	SINGLES/
...	...	...	...	...	...	...	...	...	...	...	
264829	2370701	2018-12-08	88	240378	24	Grain Waves Sweet Chilli 210g	2	7.2	210	GRNWVES	YOUNG
264830	2370751	2018-10-01	88	240394	60	Kettle Tortilla ChpsFeta&Garlic 150g	2	9.2	150	KETTLE	YOUNG
264831	2370961	2018-10-24	88	240480	70	Tyrrells Crisps Lightly Salted 165g	2	8.4	165	TYRRELLS	OLDER
264832	2370961	2018-10-27	88	240481	65	Old El Paso Salsa Dip Chnky Tom Ht300g	2	10.2	300	OLD	OLDER
264833	2373711	2018-12-14	88	241815	16	Smiths Crinkle Chips Salt & Vinegar 330g	2	11.4	330	SMITHS	SINGLES/

264834 rows × 15 columns

Monthly calculation for each store

```
totSales= df.groupby(['STORE_NBR','year_month'])['TOT_SALES'].sum().reset_index()
totSales
```

	STORE_NBR	year_month	TOT_SALES
0	1	2018-06-01	4.8
1	1	2018-07-01	220.2
2	1	2018-08-01	171.8
3	1	2018-09-01	278.2
4	1	2018-10-01	186.4
...	...	...	...
3382	272	2019-02-01	395.5
3383	272	2019-03-01	478.5
3384	272	2019-04-01	415.5
3385	272	2019-05-01	322.8
3386	272	2019-06-01	297.2

```
measureOverTime2 = pd.DataFrame(data=totSales)


nTxn= df.groupby(['STORE_NBR','year_month'])['TXN_ID'].count().reset_index(drop=True)
nTxn
```

```
0        2
1       55
2       41
3       62
4       45
       ..
3382    48
3383    58
3384    52
3385    41
3386    35
Name: TXN_ID, Length: 3387, dtype: int64
```

```
sorted(df['year_month'].unique())
```

```
[numpy.datetime64('2018-06-01T00:00:00.000000000'),
 numpy.datetime64('2018-07-01T00:00:00.000000000'),
 numpy.datetime64('2018-08-01T00:00:00.000000000'),
 numpy.datetime64('2018-09-01T00:00:00.000000000'),
 numpy.datetime64('2018-10-01T00:00:00.000000000'),
 numpy.datetime64('2018-11-01T00:00:00.000000000'),
 numpy.datetime64('2018-12-01T00:00:00.000000000'),
 numpy.datetime64('2019-01-01T00:00:00.000000000'),
 numpy.datetime64('2019-02-01T00:00:00.000000000'),
 numpy.datetime64('2019-03-01T00:00:00.000000000'),
 numpy.datetime64('2019-04-01T00:00:00.000000000'),
 numpy.datetime64('2019-05-01T00:00:00.000000000'),
 numpy.datetime64('2019-06-01T00:00:00.000000000')]
```

```
measureOverTime2['nCustomers'] = df.groupby(['STORE_NBR','year_month','LYLTY_CARD_NBR'])['DATE'].count().groupby(['STORE_NBR','yea
measureOverTime2.head()
```

	STORE_NBR	year_month	TOT_SALES	nCustomers
0	1	2018-06-01	4.8	2
1	1	2018-07-01	220.2	52
2	1	2018-08-01	171.8	41
3	1	2018-09-01	278.2	59
4	1	2018-10-01	186.4	44

```
measureOverTime2['nTxnPerCust'] = nTxn/measureOverTime2['nCustomers']
measureOverTime2.head()
```

	STORE_NBR	year_month	TOT_SALES	nCustomers	nTxnPerCust
0	1	2018-06-01	4.8	2	1.000000
1	1	2018-07-01	220.2	52	1.057692
2	1	2018-08-01	171.8	41	1.000000
3	1	2018-09-01	278.2	59	1.050847
4	1	2018-10-01	186.4	44	1.022727

```
totQty = df.groupby(['STORE_NBR','year_month'])['PROD_QTY'].sum().reset_index(drop=True)
totQty
```

```
0         2
1        65
2        53
3        75
4        57
       ...
3382     91
3383    111
3384     97
3385     73
3386     66
Name: PROD_QTY, Length: 3387, dtype: int64
```

```
measureOverTime2['nChipsPerTxn'] = totQty/nTxn
measureOverTime2
```

	STORE_NBR	year_month	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn
0	1	2018-06-01	4.8	2	1.000000	1.000000
1	1	2018-07-01	220.2	52	1.057692	1.181818
2	1	2018-08-01	171.8	41	1.000000	1.292683
3	1	2018-09-01	278.2	59	1.050847	1.209677
4	1	2018-10-01	186.4	44	1.022727	1.266667
...	...	...	...	...	...	...
3382	272	2019-02-01	395.5	45	1.066667	1.895833
3383	272	2019-03-01	478.5	52	1.115385	1.913793
3384	272	2019-04-01	415.5	50	1.040000	1.865385
3385	272	2019-05-01	322.8	36	1.138889	1.780488
3386	272	2019-06-01	297.3	32	1.093750	1.885714

3387 rows × 6 columns

```
measureOverTime2['avgPricePerUnit'] = totSales['TOT_SALES']/totQty
measureOverTime2
```

	STORE_NBR	year_month	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	2018-06-01	4.8	2	1.000000	1.000000	2.400000
1	1	2018-07-01	220.2	52	1.057692	1.181818	3.387692
2	1	2018-08-01	171.8	41	1.000000	1.292683	3.241509
3	1	2018-09-01	278.2	59	1.050847	1.209677	3.709333
4	1	2018-10-01	186.4	44	1.022727	1.266667	3.270175
...	...	...	...	...	...	...	...
3382	272	2019-02-01	395.5	45	1.066667	1.895833	4.346154
3383	272	2019-03-01	478.5	52	1.115385	1.913793	4.310811
3384	272	2019-04-01	415.5	50	1.040000	1.865385	4.283505
3385	272	2019-05-01	322.8	36	1.138889	1.780488	4.421918
3386	272	2019-06-01	297.3	32	1.093750	1.885714	4.504545

3387 rows × 7 columns

Filter pre-trial & stores with full obs

```
measureOverTime2.set_index('year_month', inplace=True)
```

```
preTrialMeasures = measureOverTime2.loc['2018-06-01':'2019-01-01'].reset_index()
preTrialMeasures
```

	year_month	STORE_NBR	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	2018-06-01	1	4.8	2	1.000000	1.000000	2.400000
1	2018-07-01	1	220.2	52	1.057692	1.181818	3.387692
2	2018-08-01	1	171.8	41	1.000000	1.292683	3.241509
3	2018-09-01	1	278.2	59	1.050847	1.209677	3.709333
4	2018-10-01	1	186.4	44	1.022727	1.266667	3.270175

Owen's *Solution*

```
measureOverTime = df.groupby(['STORE_NBR','year_month','LYLTY_CARD_NBR']).\
                     agg(
                         totSalesPerCust=('TOT_SALES', sum),
                         nTxn=('TXN_ID', "count"),
                         nChips=('PROD_QTY', sum)
                         ).\
                     groupby(['STORE_NBR','year_month']).\
                     agg(
                         totSales=("totSalesPerCust", sum),
                         nCustomers=("nTxn", "count"),
                         nTxnPerCust=("nTxn", lambda x: x.sum()/x.count()),
                         totChips=("nChips", sum),
                         totTxn=("nTxn", sum)).\
                     reset_index()


measureOverTime['nChipsPerTxn'] = measureOverTime['totChips']/measureOverTime['totTxn']
measureOverTime['avgPricePerUnit'] = measureOverTime['totSales']/measureOverTime['totChips']
measureOverTime.drop(['totChips', 'totTxn'], axis=1, inplace=True)
```

Calculate correlation

```
preTrialMeasures
```

	year_month	STORE_NBR	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	2018-06-01	1	4.8	2	1.000000	1.000000	2.400000
1	2018-07-01	1	220.2	52	1.057692	1.181818	3.387692
2	2018-08-01	1	171.8	41	1.000000	1.292683	3.241509
3	2018-09-01	1	278.2	59	1.050847	1.209677	3.709333
4	2018-10-01	1	186.4	44	1.022727	1.266667	3.270175
...	...	...	...	...	...	...	...
2062	2018-09-01	272	295.9	31	1.129032	1.971429	4.288406
2063	2018-10-01	272	438.0	45	1.155556	1.942308	4.336634
2064	2018-11-01	272	384.0	42	1.095238	1.934783	4.314607
2065	2018-12-01	272	388.7	45	1.000000	1.888889	4.572941
2066	2019-01-01	272	423.0	46	1.086957	1.920000	4.406250

2067 rows × 7 columns

```
# Input
inputTable = preTrialMeasures
metricCol = 'TOT_SALES'
storeComparison = 77

x = 1


corr = preTrialMeasures.\
          loc[preTrialMeasures['STORE_NBR'].\
          isin([x,storeComparison])].\
          loc[:, ['year_month', 'STORE_NBR', metricCol]].\
          pivot(index='year_month', columns='STORE_NBR', values=metricCol).\
          corr().\
          iloc[0, 1]


preTrialMeasures.loc[preTrialMeasures['STORE_NBR'].isin([x,storeComparison])].loc[:, ['year_month', 'STORE_NBR', metricCol]].\
pivot(index='year_month', columns='STORE_NBR', values=metricCol).corr()
```

STORE_NBR	1	77
STORE_NBR		
1	1.000000	0.841751

```
df = pd.DataFrame(columns=['Store1', 'Store2', 'corr_measure'])


df.append({'Store1':x, 'Store2':storeComparison, 'corr_measure':corr}, ignore_index=True)
```

	Store1	Store2	corr_measure
0	1.0	77.0	0.841751

```
def calculateCorrelation(inputTable, metricCol, storeComparison):
    df = pd.DataFrame(columns=['Store1', 'Store2', 'corr_measure'])
    for x in inputTable.STORE_NBR.unique():
      if x in [77, 86, 88]:
        pass
      else:
        corr = inputTable.\
                        loc[inputTable['STORE_NBR'].\
                        isin([x,storeComparison])].\
                        loc[:, ['year_month', 'STORE_NBR', metricCol]].\
                        pivot(index='year_month', columns='STORE_NBR', values=metricCol).\
                        corr().\
                        iloc[0, 1]
        df = df.append({'Store1':storeComparison, 'Store2':x, 'corr_measure':corr}, ignore_index=True)
    return(df)


calcCorrTable = calculateCorrelation(inputTable=preTrialMeasures, metricCol='nCustomers', storeComparison=77)


calcCorrTable
```

	Store1	Store2	corr_measure
0	77.0	1.0	0.909962
1	77.0	2.0	0.881730
2	77.0	3.0	0.975036
3	77.0	4.0	0.916797
4	77.0	5.0	0.962518
...	...	...	...
263	77.0	268.0	0.913429
264	77.0	269.0	0.925431
265	77.0	270.0	0.895506
266	77.0	271.0	0.943397
267	77.0	272.0	0.904827

268 rows × 3 columns

Calculate magnitude distance

```
inputTable = preTrialMeasures
metricCol = 'TOT_SALES'
storeComparison = '77'

x='2'


mag = preTrialMeasures.\
          loc[preTrialMeasures['STORE_NBR'].isin([x, storeComparison])].\
          loc[:, ['year_month', 'STORE_NBR', metricCol]].\
          pivot(index='year_month', columns='STORE_NBR', values=metricCol).\
          reset_index().rename_axis(None, axis=1)
mag
```

	year_month	2	77
0	2018-06-01	12.1	15.6
1	2018-07-01	145.2	289.1
2	2018-08-01	191.9	247.6

```
mag.columns = mag.columns.map(str)
mag
```

	year_month	2	77
0	2018-06-01	12.1	15.6
1	2018-07-01	145.2	289.1
2	2018-08-01	191.9	247.6
3	2018-09-01	152.8	225.2
4	2018-10-01	170.1	204.5
5	2018-11-01	163.5	247.7
6	2018-12-01	133.1	270.1
7	2019-01-01	166.1	210.2

```
mag['measures'] = mag.apply(lambda row: row[x]-row[storeComparison], axis=1).abs()

mag
```

	year_month	2	77	measures
0	2018-06-01	12.1	15.6	3.5
1	2018-07-01	145.2	289.1	143.9
2	2018-08-01	191.9	247.6	55.7
3	2018-09-01	152.8	225.2	72.4
4	2018-10-01	170.1	204.5	34.4
5	2018-11-01	163.5	247.7	84.2
6	2018-12-01	133.1	270.1	137.0
7	2019-01-01	166.1	210.2	44.1

```
mag['Store1'] = x
mag['Store2'] = storeComparison


df_temp = mag.loc[:, ['Store1', 'Store2', 'year_month','measures']]


df_temp
```

	Store1	Store2	year_month	measures
0	2	77	2018-06-01	3.5
1	2	77	2018-07-01	143.9
2	2	77	2018-08-01	55.7
3	2	77	2018-09-01	72.4
4	2	77	2018-10-01	34.4
5	2	77	2018-11-01	84.2
6	2	77	2018-12-01	137.0
7	2	77	2019-01-01	44.1

```
df = pd.DataFrame(columns=['Store1', 'Store2', 'year_month','measures'])
df
```

	Store1	Store2	year_month	measures

```
inputTable = preTrialMeasures
metricCol = 'TOT_SALES'
storeComparison = '77'
df = pd.DataFrame(columns=['Store1', 'Store2', 'year_month','measures'])
for x in inputTable.STORE_NBR.unique():
      if x in [77, 86, 88]:
        pass
      else:
```

```
      mag = preTrialMeasures.\
        loc[preTrialMeasures['STORE_NBR'].\
        isin([x, storeComparison])].\
        loc[:, ['year_month', 'STORE_NBR', metricCol]].\
        pivot(index='year_month', columns='STORE_NBR', values=metricCol).\
        reset_index().rename_axis(None, axis=1)
      mag.columns = ['year_month', 'Store1', 'Store2']
      mag['measures'] = mag.apply(lambda row: row['Store1']-row['Store2'], axis=1).abs()
      mag['Store1'] = x
      mag['Store2'] = storeComparison
      df_temp = mag.loc[:, ['Store1', 'Store2', 'year_month','measures']]
      df = pd.concat([df, df_temp])


df
```

	Store1	Store2	year_month	measures
0	1	77	2018-06-01	10.8
1	1	77	2018-07-01	68.9
2	1	77	2018-08-01	75.8
3	1	77	2018-09-01	53.0
4	1	77	2018-10-01	18.1
...	...	...	...	...
3	272	77	2018-09-01	70.7
4	272	77	2018-10-01	233.5
5	272	77	2018-11-01	136.3
6	272	77	2018-12-01	118.6
7	272	77	2019-01-01	212.8

2144 rows × 4 columns

```
def calculateMagnitudeDistance(inputTable, metricCol, storeComparison):
  df = pd.DataFrame(columns=['Store1', 'Store2', 'year_month','measures'])
  for x in inputTable.STORE_NBR.unique():
      if x in [77, 86, 88]:
        pass
      else:
        mag = preTrialMeasures.\
          loc[preTrialMeasures['STORE_NBR'].\
          isin([x, storeComparison])].\
          loc[:, ['year_month', 'STORE_NBR', metricCol]].\
          pivot(index='year_month', columns='STORE_NBR', values=metricCol).\
          reset_index().rename_axis(None, axis=1)
        mag.columns = ['year_month', 'Store1', 'Store2']
        mag['measures'] = mag.apply(lambda row: row['Store1']-row['Store2'], axis=1).abs()
        mag['Store1'] = storeComparison
        mag['Store2'] = x
        df_temp = mag.loc[:, ['Store1', 'Store2', 'year_month','measures']]
        df = pd.concat([df, df_temp])
  return df


def finalDistTable(inputTable, metricCol, storeComparison):
      calcDistTable = calculateMagnitudeDistance(inputTable, metricCol, storeComparison)
      minMaxDist = calcDistTable.groupby(['Store1','year_month'])['measures'].agg(['max','min']).reset_index()
      distTable = calcDistTable.merge(minMaxDist, on=['year_month', 'Store1'])
      distTable['magnitudeMeasure']= distTable.apply(lambda row: 1- (row['measures']-row['min'])/(row['max']-row['min']),axis=1)
      finalDistTable = distTable.groupby(['Store1','Store2'])['magnitudeMeasure'].mean().reset_index()
      finalDistTable.columns = ['Store1','Store2','mag_measure']
      return finalDistTable


calcDistTable = calculateMagnitudeDistance(inputTable=preTrialMeasures, metricCol='nCustomers', storeComparison='77')
calcDistTable
```

	Store1	Store2	year_month	measures
0	77	1	2018-06-01	0
1	77	1	2018-07-01	2
2	77	1	2018-08-01	4
3	77	1	2018-09-01	17

Standardise the magnitude distance

```
#calcDistTable.groupby(['Store1','year_month'])['measures'].apply(lambda g: g.max() - g.min()).reset_index()
```

4	77	272	2018-10-01	8

```
minMaxDist = calcDistTable.groupby(['Store1','year_month'])['measures'].agg(['max','min']).reset_index()
minMaxDist
```

	Store1	year_month	max	min
0	77	2018-06-01	8	0.0
1	77	2018-07-01	91	0.0
2	77	2018-08-01	92	0.0
3	77	2018-09-01	96	0.0
4	77	2018-10-01	102	0.0
5	77	2018-11-01	98	0.0
6	77	2018-12-01	105	0.0
7	77	2019-01-01	100	0.0

```
calcDistTable.merge(minMaxDist, on=['year_month', 'Store1'])
```

	Store1	Store2	year_month	measures	max	min
0	77	1	2018-06-01	0	8	0.0
1	77	2	2018-06-01	0	8	0.0
2	77	3	2018-06-01	1	8	0.0
3	77	4	2018-06-01	3	8	0.0
4	77	5	2018-06-01	0	8	0.0
...	...	...	...	...	...	...
2139	77	268	2019-01-01	1	100	0.0
2140	77	269	2019-01-01	71	100	0.0
2141	77	270	2019-01-01	82	100	0.0
2142	77	271	2019-01-01	57	100	0.0
2143	77	272	2019-01-01	10	100	0.0

2144 rows × 6 columns

```
distTable = calcDistTable.merge(minMaxDist, on=['year_month', 'Store1'])
distTable
```

	Store1	Store2	year_month	measures	max	min
0	77	1	2018-06-01	0	8	0.0
1	77	2	2018-06-01	0	8	0.0
2	77	3	2018-06-01	1	8	0.0
3	77	4	2018-06-01	3	8	0.0
4	77	5	2018-06-01	0	8	0.0
...	...	...	...	...	...	...
2139	77	268	2019-01-01	1	100	0.0
2140	77	269	2019-01-01	71	100	0.0
2141	77	270	2019-01-01	82	100	0.0
2142	77	271	2019-01-01	57	100	0.0
2143	77	272	2019-01-01	10	100	0.0

2144 rows × 6 columns

```
distTable['magnitudeMeasure']= distTable.apply(lambda row: 1- (row['measures']-row['min'])/(row['max']-row['min']),axis=1)
```

```
distTable
```

	Store1	Store2	year_month	measures	max	min	magnitudeMeasure
0	77	1	2018-06-01	0	8	0.0	1.000
1	77	2	2018-06-01	0	8	0.0	1.000
2	77	3	2018-06-01	1	8	0.0	0.875
3	77	4	2018-06-01	3	8	0.0	0.625
4	77	5	2018-06-01	0	8	0.0	1.000
...	...	...	...	...	...	...	...
2139	77	268	2019-01-01	1	100	0.0	0.990
2140	77	269	2019-01-01	71	100	0.0	0.290
2141	77	270	2019-01-01	82	100	0.0	0.180
2142	77	271	2019-01-01	57	100	0.0	0.430
2143	77	272	2019-01-01	10	100	0.0	0.900

2144 rows × 7 columns

Merge nTotSals & nCustomers

```
corr_nSales = calculateCorrelation(inputTable=preTrialMeasures, metricCol='TOT_SALES',storeComparison='77')
corr_nSales
```

	Store1	Store2	corr_measure
0	77	1	0.841751
1	77	2	0.840647
2	77	3	0.967552
3	77	4	0.895463
4	77	5	0.935511
...	...	...	...
263	77	268	0.845355
264	77	269	0.889074
265	77	270	0.927951
266	77	271	0.931167
267	77	272	0.891416

268 rows × 3 columns

```
corr_nCustomers = calculateCorrelation(inputTable=preTrialMeasures, metricCol='nCustomers',storeComparison='77')
corr_nCustomers
```

	Store1	Store2	corr_measure
0	77	1	0.909962
1	77	2	0.881730
2	77	3	0.975036
3	77	4	0.916797
4	77	5	0.962518
...	...	...	...
263	77	268	0.913429
264	77	269	0.925431
265	77	270	0.895506
266	77	271	0.943397
267	77	272	0.904827

268 rows × 3 columns

```
magnitude_nSales = finalDistTable(inputTable=preTrialMeasures, metricCol='TOT_SALES',storeComparison='77')
magnitude_nSales
```

	Store1	Store2	mag_measure
0	77	1	0.938089
1	77	2	0.939103
2	77	3	0.417810
3	77	4	0.225170
4	77	5	0.609797
...	...	...	...
263	77	268	0.945110
264	77	269	0.516665
265	77	270	0.397266
266	77	271	0.535115

```
magnitude_nCustomers = finalDistTable(inputTable=preTrialMeasures, metricCol='nCustomers',storeComparison='77')
magnitude_nCustomers
```

	Store1	Store2	mag_measure
0	77	1	0.950075
1	77	2	0.933215
2	77	3	0.406144
3	77	4	0.243139
4	77	5	0.541307
...	...	...	...
263	77	268	0.935928
264	77	269	0.417467
265	77	270	0.315052
266	77	271	0.488128
267	77	272	0.939719

268 rows × 3 columns

Get control store

```
score_nSales = corr_nSales.merge(magnitude_nSales, on=['Store1','Store2'])
score_nSales['scoreNSales'] = score_nSales.apply(lambda row: row['corr_measure']*0.5 + row['mag_measure']*0.5, axis=1)
score_nSales = score_nSales.loc[:,['Store1','Store2', 'scoreNSales']]
score_nSales
```

	Store1	Store2	scoreNSales
0	77	1	0.889920
1	77	2	0.889875
2	77	3	0.692681
3	77	4	0.560317
4	77	5	0.772654
...	...	...	...
263	77	268	0.895233
264	77	269	0.702870
265	77	270	0.662609
266	77	271	0.733141
267	77	272	0.888291

268 rows × 3 columns

```
score_nCustomers = corr_nCustomers.merge(magnitude_nCustomers, on=['Store1','Store2'])
score_nCustomers['scoreNCust'] = score_nCustomers.apply(lambda row: row['corr_measure']*0.5 + row['mag_measure']*0.5, axis=1)
score_nCustomers = score_nCustomers.loc[:,['Store1','Store2','scoreNCust']]
score_nCustomers
```

	Store1	Store2	scoreNCust
0	77	1	0.930018
1	77	2	0.907473
2	77	3	0.690590
3	77	4	0.579968
4	77	5	0.751912
...	...	...	...
263	77	268	0.924679
264	77	269	0.671449
265	77	270	0.605279

```
score_Control = score_nSales.merge(score_nCustomers, on=['Store1','Store2'])
score_Control
```

	Store1	Store2	scoreNSales	scoreNCust
0	77	1	0.889920	0.930018
1	77	2	0.889875	0.907473
2	77	3	0.692681	0.690590
3	77	4	0.560317	0.579968
4	77	5	0.772654	0.751912
...	...	...	...	...
263	77	268	0.895233	0.924679
264	77	269	0.702870	0.671449
265	77	270	0.662609	0.605279
266	77	271	0.733141	0.715762
267	77	272	0.888291	0.922273

268 rows × 4 columns

```
score_Control['finalControlScore'] = score_Control.apply(lambda row: row['scoreNSales']*0.5 + row['scoreNCust']*0.5, axis=1)
score_Control
```

	Store1	Store2	scoreNSales	scoreNCust	finalControlScore
0	77	1	0.889920	0.930018	0.909969
1	77	2	0.889875	0.907473	0.898674
2	77	3	0.692681	0.690590	0.691635
3	77	4	0.560317	0.579968	0.570142
4	77	5	0.772654	0.751912	0.762283
...	...	...	...	...	...
263	77	268	0.895233	0.924679	0.909956
264	77	269	0.702870	0.671449	0.687159
265	77	270	0.662609	0.605279	0.633944
266	77	271	0.733141	0.715762	0.724452
267	77	272	0.888291	0.922273	0.905282

268 rows × 5 columns

```
final_control_store = score_Control['finalControlScore'].max()
```

```
score_Control[score_Control['finalControlScore']==final_control_store]
```

	Store1	Store2	scoreNSales	scoreNCust	finalControlScore
228	77	233	0.962765	0.981021	0.971893

Visualization the control store

```
measureOverTime['Store_type'] = measureOverTime.apply(lambda row: 'Trail' if row['STORE_NBR']==77 else ('Control' if row['STORE_NI
measureOverTime
```

	STORE_NBR	year_month	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	Store_type
0	1	2018-06-01	4.8	2	1.000000	1.000000	2.400000	Other stores
1	1	2018-07-01	220.2	52	1.057692	1.181818	3.387692	Other stores
2	1	2018-08-01	171.8	41	1.000000	1.292683	3.241509	Other stores
3	1	2018-09-01	278.2	59	1.050847	1.209677	3.709333	Other stores
4	1	2018-10-01	186.4	44	1.022727	1.266667	3.270175	Other stores
...	...	...	...	...	...	...	...	...
3382	272	2019-02-01	395.5	45	1.066667	1.895833	4.346154	Other stores
3383	272	2019-03-01	478.5	52	1.115385	1.913793	4.310811	Other stores
3384	272	2019-04-01	415.5	50	1.040000	1.865385	4.283505	Other stores
3385	272	2019-05-01	322.8	36	1.138889	1.780488	4.421918	Other stores
3386	272	2019-06-01	297.3	32	1.093750	1.885714	4.504545	Other stores

```
measureOverTime['Store_type'].unique()
```

```
array(['Other stores', 'Trail', 'Control'], dtype=object)
```

```
measureOverTimeSales = measureOverTime.groupby(['year_month','Store_type'])['totSales'].mean().reset_index()
measureOverTimeSales
```

	year_month	Store_type	totSales
0	2018-06-01	Control	6.600000
1	2018-06-01	Other stores	24.654378
2	2018-06-01	Trail	15.600000
3	2018-07-01	Control	286.200000

```
measureOverTimeSales.set_index('year_month',inplace=True)
```

5	2018-07-01	Trail	289.100000

```
pastSales = measureOverTimeSales.loc['2018-06-01':'2019-01-01'].reset_index()
pastSales
```

	year_month	Store_type	totSales
0	2018-06-01	Control	6.600000
1	2018-06-01	Other stores	24.654378
2	2018-06-01	Trail	15.600000
3	2018-07-01	Control	286.200000
4	2018-07-01	Other stores	623.767803
5	2018-07-01	Trail	289.100000
6	2018-08-01	Control	300.200000
7	2018-08-01	Other stores	601.354771
8	2018-08-01	Trail	247.600000
9	2018-09-01	Control	228.000000
10	2018-09-01	Other stores	612.650192
11	2018-09-01	Trail	225.200000
12	2018-10-01	Control	194.100000
13	2018-10-01	Other stores	624.620532
14	2018-10-01	Trail	204.500000
15	2018-11-01	Control	209.400000
16	2018-11-01	Other stores	609.712214
17	2018-11-01	Trail	247.700000
18	2018-12-01	Control	262.000000
19	2018-12-01	Other stores	639.750192
20	2018-12-01	Trail	270.100000
21	2019-01-01	Control	183.700000
22	2019-01-01	Other stores	623.397318
23	2019-01-01	Trail	210.200000
32	2019-04-01	Trail	245.200000

```
px.line(data_frame=pastSales, x='year_month', y='totSales', color='Store_type', title='Total sales by month',labels={'year_month':
```

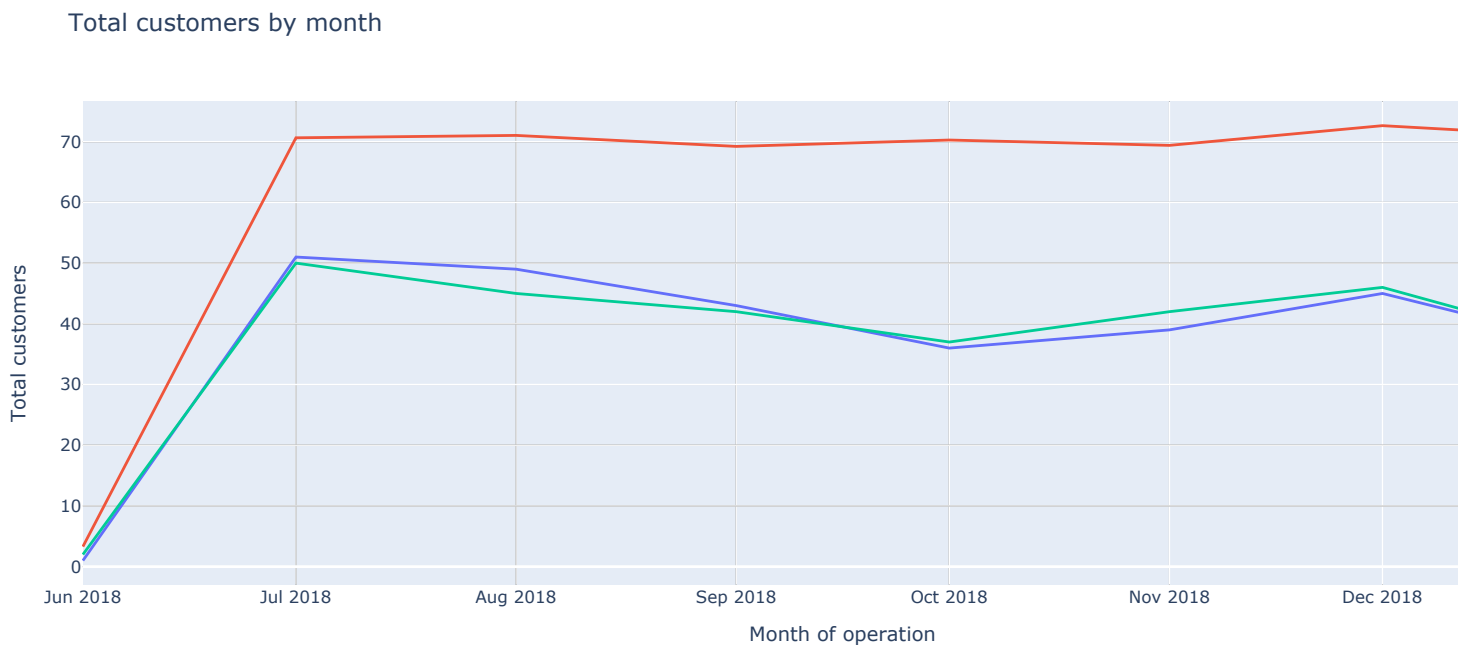
```
measureOverTimeCusts = measureOverTime.groupby(['year_month','Store_type'])['nCustomers'].mean().reset_index()
measureOverTimeCusts
```

	year_month	Store_type	nCustomers
0	2018-06-01	Control	1.000000
1	2018-06-01	Other stores	3.304147
2	2018-06-01	Trail	2.000000
3	2018-07-01	Control	51.000000
4	2018-07-01	Other stores	70.640152
5	2018-07-01	Trail	50.000000
6	2018-08-01	Control	49.000000
7	2018-08-01	Other stores	71.034351
8	2018-08-01	Trail	45.000000
9	2018-09-01	Control	43.000000
10	2018-09-01	Other stores	69.222222
11	2018-09-01	Trail	42.000000
12	2018-10-01	Control	36.000000
13	2018-10-01	Other stores	70.273764
14	2018-10-01	Trail	37.000000
15	2018-11-01	Control	39.000000
16	2018-11-01	Other stores	69.393130
17	2018-11-01	Trail	42.000000
18	2018-12-01	Control	45.000000
19	2018-12-01	Other stores	72.639847
20	2018-12-01	Trail	46.000000
21	2019-01-01	Control	36.000000
22	2019-01-01	Other stores	70.613027
23	2019-01-01	Trail	36.000000
24	2019-02-01	Control	46.000000
25	2019-02-01	Other stores	65.347328
26	2019-02-01	Trail	46.000000
27	2019-03-01	Control	38.000000
28	2019-03-01	Other stores	71.380228
29	2019-03-01	Trail	51.000000
30	2019-04-01	Control	36.000000
31	2019-04-01	Other stores	69.179389
32	2019-04-01	Trail	45.000000
33	2019-05-01	Control	51.000000
34	2019-05-01	Other stores	70.919540
35	2019-05-01	Trail	55.000000
36	2019-06-01	Control	41.000000
37	2019-06-01	Other stores	67.396947

```
measureOverTimeCusts.set_index('year_month',inplace=True)
pastCustomers = measureOverTimeCusts.loc['2018-06-01':'2019-01-01'].reset_index()
pastCustomers
```

	year_month	Store_type	nCustomers
0	2018-06-01	Control	1.000000
1	2018-06-01	Other stores	3.304147
2	2018-06-01	Trail	2.000000
3	2018-07-01	Control	51.000000
4	2018-07-01	Other stores	70.640152
5	2018-07-01	Trail	50.000000
6	2018-08-01	Control	49.000000
7	2018-08-01	Other stores	71.034351
8	2018-08-01	Trail	45.000000
9	2018-09-01	Control	43.000000
10	2018-09-01	Other stores	69.222222
11	2018-09-01	Trail	42.000000
12	2018-10-01	Control	36.000000
13	2018-10-01	Other stores	70.273764
14	2018-10-01	Trail	37.000000
15	2018-11-01	Control	39.000000
16	2018-11-01	Other stores	69.393130
17	2018-11-01	Trail	42.000000

```
px.line(data_frame=pastCustomers, x='year_month', y='nCustomers', color='Store_type', title='Total customers by month',labels={'ye
```



Assessment of trial period

Calculate for totSales

Scale sales

```
preTrialMeasures
```

	year_month	STORE_NBR	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	2018-06-01	1	4.8	2	1.000000	1.000000	2.400000
1	2018-07-01	1	220.2	52	1.057692	1.181818	3.387692
2	2018-08-01	1	171.8	41	1.000000	1.292683	3.241509
3	2018-09-01	1	278.2	59	1.050847	1.209677	3.709333
4	2018-10-01	1	186.4	44	1.022727	1.266667	3.270175
...	...	...	...	...	...	...	...
2062	2018-09-01	272	295.9	31	1.129032	1.971429	4.288406

```
preTrialMeasures.loc[preTrialMeasures['STORE_NBR']==77, 'TOT_SALES'].sum()
```

```
1710.0
```

```
preTrialMeasures.loc[preTrialMeasures['STORE_NBR']==233, 'TOT_SALES'].sum()
```

```
1670.2
```

```
scalingFactorForControlSales = preTrialMeasures.loc[preTrialMeasures['STORE_NBR']==77, 'TOT_SALES'].sum() / preTrialMeasures.loc[p
scalingFactorForControlSales
```

```
1.0238294814992217
```

Apply the scaling factor

```
scaledControlSales = measureOverTimeSales.loc[measureOverTimeSales['Store_type']=='Control','totSales'].reset_index()
scaledControlSales
```

	year_month	totSales
0	2018-06-01	6.6
1	2018-07-01	286.2
2	2018-08-01	300.2
3	2018-09-01	228.0
4	2018-10-01	194.1
5	2018-11-01	209.4
6	2018-12-01	262.0
7	2019-01-01	183.7
8	2019-02-01	244.7
9	2019-03-01	193.2
10	2019-04-01	181.8
11	2019-05-01	316.0
12	2019-06-01	221.0

```
scaledControlSales['scaledControlSales'] = scaledControlSales.apply(lambda row: row['totSales']*scalingFactorForControlSales,axis=
scaledControlSales
```

	year_month	totSales	scaledControlSales
0	2018-06-01	6.6	6.757275
1	2018-07-01	286.2	293.019998
2	2018-08-01	300.2	307.353610
3	2018-09-01	228.0	233.433122
4	2018-10-01	194.1	198.725302
5	2018-11-01	209.4	214.389893
6	2018-12-01	262.0	268.243324
7	2019-01-01	183.7	188.077476
8	2019-02-01	244.7	250.531074
9	2019-03-01	193.2	197.803856
10	2019-04-01	181.8	186.132200
11	2019-05-01	316.0	323.530116
12	2019-06-01	221.0	226.266315

```
TrailStoreSales = measureOverTimeSales.loc[measureOverTimeSales['Store_type']=='Trail',['totSales']]
TrailStoreSales
```

year_month	totSales
2018-06-01	15.6
2018-07-01	289.1
2018-08-01	247.6
2018-09-01	225.2
2018-10-01	204.5
2018-11-01	247.7
2018-12-01	270.1
2019-01-01	210.2
2019-02-01	240.5
2019-03-01	283.6
2019-04-01	245.2
2019-05-01	307.4
2019-06-01	253.3

```
TrailStoreSales.columns = ['trailSales']
TrailStoreSales
```

year_month	trailSales
2018-06-01	15.6
2018-07-01	289.1
2018-08-01	247.6
2018-09-01	225.2
2018-10-01	204.5
2018-11-01	247.7
2018-12-01	270.1
2019-01-01	210.2
2019-02-01	240.5
2019-03-01	283.6
2019-04-01	245.2
2019-05-01	307.4
2019-06-01	253.3

%Diff between scaled control and trial for sales

```
percentageDiff = scaledControlSales.merge(TrailStoreSales, on='year_month',)
percentageDiff
```

	year_month	totSales	scaledControlSales	trailSales

```
percentageDiff['percentDiff'] = percentageDiff.apply(lambda row: (row['scaledControlSales']-row['trailSales'])/row['scaledControl
```

```
percentageDiff
```

	year_month	totSales	scaledControlSales	trailSales	percentDiff
0	2018-06-01	6.6	6.757275	15.6	-1.308623
1	2018-07-01	286.2	293.019998	289.1	0.013378
2	2018-08-01	300.2	307.353610	247.6	0.194413
3	2018-09-01	228.0	233.433122	225.2	0.035270
4	2018-10-01	194.1	198.725302	204.5	-0.029059
5	2018-11-01	209.4	214.389893	247.7	-0.155372
6	2018-12-01	262.0	268.243324	270.1	-0.006922
7	2019-01-01	183.7	188.077476	210.2	-0.117625
8	2019-02-01	244.7	250.531074	240.5	0.040039
9	2019-03-01	193.2	197.803856	283.6	-0.433744
10	2019-04-01	181.8	186.132200	245.2	-0.317343
11	2019-05-01	316.0	323.530116	307.4	0.049857
12	2019-06-01	221.0	226.266315	253.3	-0.119477

Get standard deviation

```
stdDev = percentageDiff.loc[percentageDiff['year_month']< '2019-02-01', 'percentDiff'].std(ddof=8-1)
stdDev
```

```
1.2467607632480449
```

Calculate the t-values for the trial months

```
from scipy.stats import ttest_ind
```

```
control = percentageDiff.loc[percentageDiff['year_month']>'2019-01-01',['scaledControlSales']]
control
```

	scaledControlSales
8	250.531074
9	197.803856
10	186.132200
11	323.530116
12	226.266315

```
trail = percentageDiff.loc[percentageDiff['year_month']>'2019-01-01',['trailSales']]
trail
```

	trailSales
8	240.5
9	283.6
10	245.2
11	307.4
12	253.3

```
ttest_ind(control,trail)
```

```
Ttest_indResult(statistic=array([-1.05806925]), pvalue=array([0.3209233]))
```

The null hypothesis here is "the sales between control and trial stores has **NO** significantly difference in trial period." The pvalue is 0.32, which is 32% that they are same in sales,which is much greater than 5%. Fail to reject the null hypothesis. Therefore, we are not confident to say "the trial period impact trial store sales."

```
percentageDiff['t-value'] = percentageDiff.apply(lambda row: (row['percentDiff']- 0) / stdDev,axis=1)
percentageDiff
```

	year_month	totSales	scaledControlSales	trailSales	percentDiff	t-value
0	2018-06-01	6.6	6.757275	15.6	-1.308623	-1.049618
1	2018-07-01	286.2	293.019998	289.1	0.013378	0.010730
2	2018-08-01	300.2	307.353610	247.6	0.194413	0.155935
3	2018-09-01	228.0	233.433122	225.2	0.035270	0.028289
4	2018-10-01	194.1	198.725302	204.5	-0.029059	-0.023307
5	2018-11-01	209.4	214.389893	247.7	-0.155372	-0.124620
6	2018-12-01	262.0	268.243324	270.1	-0.006922	-0.005552
7	2019-01-01	183.7	188.077476	210.2	-0.117625	-0.094344
8	2019-02-01	244.7	250.531074	240.5	0.040039	0.032115
9	2019-03-01	193.2	197.803856	283.6	-0.433744	-0.347896
10	2019-04-01	181.8	186.132200	245.2	-0.317343	-0.254534
11	2019-05-01	316.0	323.530116	307.4	0.049857	0.039989
12	2019-06-01	221.0	226.266315	253.3	-0.119477	-0.095830

We can observe that the t-value is much larger than the 95th percentile value of the t-distribution for March and April. i.e. the increase in sales in the trial store in March and April is statistically greater than in the control store.

95th & 5th percentile of control store

```
measureOverTimeSales
```

year_month	Store_type	totSales
2018-06-01	Control	6.600000
2018-06-01	Other stores	24.654378
2018-06-01	Trail	15.600000
2018-07-01	Control	286.200000
2018-07-01	Other stores	623.767803
2018-07-01	Trail	289.100000
2018-08-01	Control	300.200000
2018-08-01	Other stores	601.354771
2018-08-01	Trail	247.600000

```
pastSales_Controls95 = measureOverTimeSales.loc[measureOverTimeSales['Store_type']=='Control']
pastSales_Controls95['totSales'] = pastSales_Controls95.apply(lambda row: row['totSales']*(1+stdDev*2),axis=1)
pastSales_Controls95.iloc[0:13,0] = 'Control 95th % confidence interval'
pastSales_Controls95.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
```

	year_month	Store_type	totSales
0	2018-06-01	Control 95th % confidence interval	23.057242
1	2018-07-01	Control 95th % confidence interval	999.845861
2	2018-08-01	Control 95th % confidence interval	1048.755162
3	2018-09-01	Control 95th % confidence interval	796.522908
4	2018-10-01	Control 95th % confidence interval	678.092528
5	2018-11-01	Control 95th % confidence interval	731.543408
6	2018-12-01	Control 95th % confidence interval	915.302640
7	2019-01-01	Control 95th % confidence interval	641.759904
8	2019-02-01	Control 95th % confidence interval	854.864718
9	2019-03-01	Control 95th % confidence interval	674.948359
10	2019-04-01	Control 95th % confidence interval	635.122214
11	2019-05-01	Control 95th % confidence interval	1103.952802
12	2019-06-01	Control 95th % confidence interval	772.068257
2019-05-01		Trail	307.400000

```
pastSales_Controls5 = measureOverTimeSales.loc[measureOverTimeSales['Store_type']=='Control']
pastSales_Controls5['totSales'] = pastSales_Controls95.apply(lambda row: row['totSales']*(1-stdDev*2),axis=1)
pastSales_Controls5.iloc[0:13,0] = 'Control 5th % confidence interval'
pastSales_Controls5.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
```

	year_month	Store_type	totSales
0	2018-06-01	Control 5th % confidence interval	-34.436487
1	2018-07-01	Control 5th % confidence interval	-1493.291316
2	2018-08-01	Control 5th % confidence interval	-1566.338411

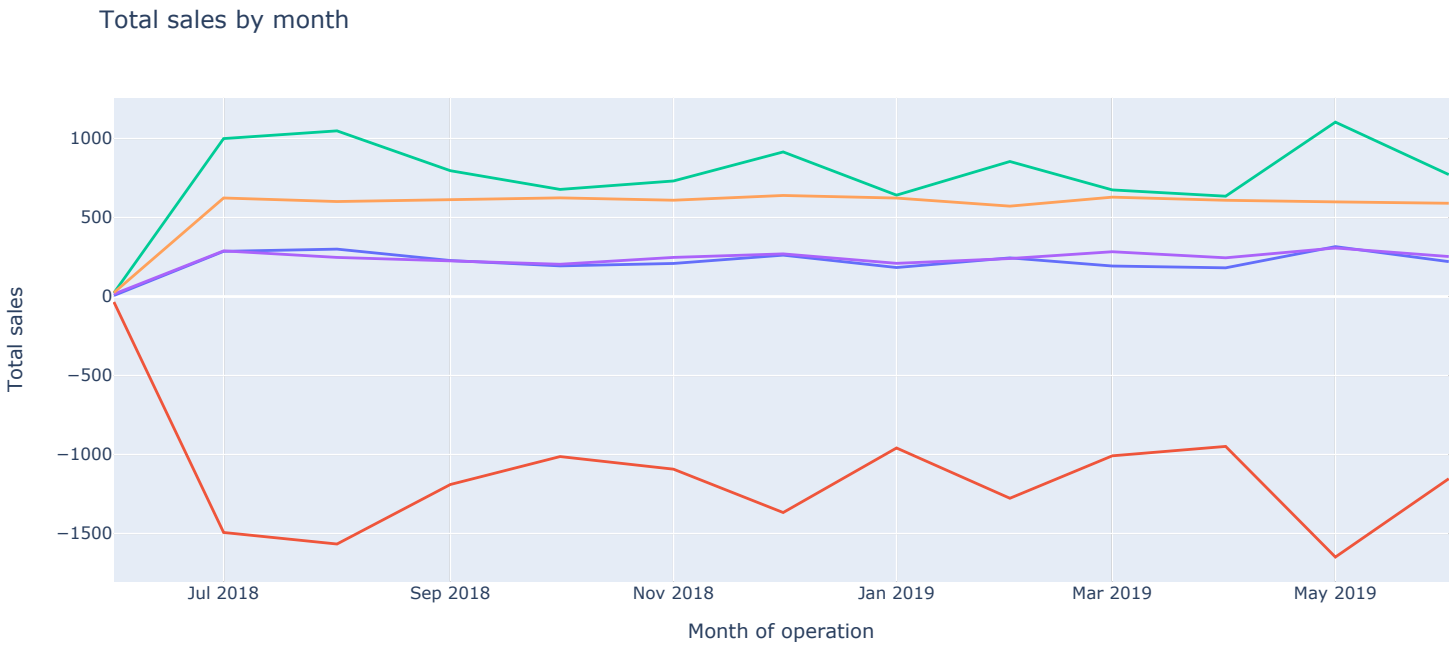
```
trialAssessment = pd.concat([measureOverTimeSales,pastSales_Controls5,pastSales_Controls95])
trialAssessment = trialAssessment.sort_values(by=['year_month'])
trialAssessment = trialAssessment.reset_index()
trialAssessment
```

	year_month	Store_type	totSales
0	2018-06-01	Control	6.600000
1	2018-06-01	Control 5th % confidence interval	-34.436487
2	2018-06-01	Control 95th % confidence interval	23.057242
3	2018-06-01	Trail	15.600000
4	2018-06-01	Other stores	24.654378
...	...	...	...
60	2019-06-01	Trail	253.300000
61	2019-06-01	Other stores	590.437405
62	2019-06-01	Control	221.000000
63	2019-06-01	Control 5th % confidence interval	-1153.100562
64	2019-06-01	Control 95th % confidence interval	772.068257

65 rows × 3 columns

Visualization Trial

```
px.line(data_frame=trialAssessment, x='year_month', y='totSales', color='Store_type', title='Total sales by month',labels={'year_r
```



Calculate for nCustomers

Scales nCustomers

```
preTrialMeasures
```

	year_month	STORE_NBR	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	2018-06-01	1	4.8	2	1.000000	1.000000	2.400000
1	2018-07-01	1	220.2	52	1.057692	1.181818	3.387692
2	2018-08-01	1	171.8	41	1.000000	1.292683	3.241509
3	2018-09-01	1	278.2	59	1.050847	1.209677	3.709333
4	2018-10-01	1	186.4	44	1.022727	1.266667	3.270175
...	...	...	...	...	...	...	...
2062	2018-09-01	272	295.9	31	1.129032	1.971429	4.288406
2063	2018-10-01	272	438.0	45	1.155556	1.942308	4.336634
2064	2018-11-01	272	384.0	42	1.095238	1.934783	4.314607
2065	2018-12-01	272	388.7	45	1.000000	1.888889	4.572941
2066	2019-01-01	272	423.0	46	1.086957	1.920000	4.406250

2067 rows × 7 columns

```
preTrialMeasures.loc[preTrialMeasures['STORE_NBR']==77,'nCustomers'].sum()
```

```
300
```

```
preTrialMeasures.loc[preTrialMeasures['STORE_NBR']==233,'nCustomers'].sum()
```

```
300
```

```
scalingFactorForControlnCustomers = preTrialMeasures.loc[preTrialMeasures['STORE_NBR']==77,'nCustomers'].sum() / preTrialMeasures.
scalingFactorForControlnCustomers
```

```
1.0
```

Apply the scaling factor

```
measureOverTime
```

	STORE_NBR	year_month	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	Store_type
0	1	2018-06-01	4.8	2	1.000000	1.000000	2.400000	Other stores
1	1	2018-07-01	220.2	52	1.057692	1.181818	3.387692	Other stores
2	1	2018-08-01	171.8	41	1.000000	1.292683	3.241509	Other stores
3	1	2018-09-01	278.2	59	1.050847	1.209677	3.709333	Other stores
4	1	2018-10-01	186.4	44	1.022727	1.266667	3.270175	Other stores
...	...	...	...	...	...	...	...	...
3382	272	2019-02-01	395.5	45	1.066667	1.895833	4.346154	Other stores
3383	272	2019-03-01	478.5	52	1.115385	1.913793	4.310811	Other stores
3384	272	2019-04-01	415.5	50	1.040000	1.865385	4.283505	Other stores
3385	272	2019-05-01	322.8	36	1.138889	1.780488	4.421918	Other stores
3386	272	2019-06-01	297.3	32	1.093750	1.885714	4.504545	Other stores

3387 rows × 8 columns

```
scaledControlNcustomers = measureOverTime.loc[measureOverTime['Store_type']=='Control',['year_month','nCustomers']]
scaledControlNcustomers
```

	year_month	nCustomers
2886	2018-06-01	1
2887	2018-07-01	51
2888	2018-08-01	49
2889	2018-09-01	43
2890	2018-10-01	36
2891	2018-11-01	39
2892	2018-12-01	45

```
scaledControlNcustomers['scaledControlNcus'] = scaledControlNcustomers.apply(lambda row: row['nCustomers']*scalingFactorForControl
scaledControlNcustomers
```

	year_month	nCustomers	scaledControlNcus
2886	2018-06-01	1	1.0
2887	2018-07-01	51	51.0
2888	2018-08-01	49	49.0
2889	2018-09-01	43	43.0
2890	2018-10-01	36	36.0
2891	2018-11-01	39	39.0
2892	2018-12-01	45	45.0
2893	2019-01-01	36	36.0
2894	2019-02-01	46	46.0
2895	2019-03-01	38	38.0
2896	2019-04-01	36	36.0
2897	2019-05-01	51	51.0
2898	2019-06-01	41	41.0

%Diff between scaled control & trail for nCustomers

```
measureOverTime.loc[measureOverTime['Store_type']=='Trail',['year_month','nCustomers']]
```

	year_month	nCustomers
945	2018-06-01	2
946	2018-07-01	50
947	2018-08-01	45
948	2018-09-01	42
949	2018-10-01	37
950	2018-11-01	42
951	2018-12-01	46
952	2019-01-01	36
953	2019-02-01	46
954	2019-03-01	51
955	2019-04-01	45
956	2019-05-01	55
957	2019-06-01	40

```
percentageDiff = scaledControlNcustomers.merge(measureOverTime.loc[measureOverTime['Store_type']=='Trail',['year_month','nCustomer
percentageDiff
```

	year_month	nCustomers_x	scaledControlNcus	nCustomers_y
0	2018-06-01	1	1.0	2
1	2018-07-01	51	51.0	50
2	2018-08-01	49	49.0	45
3	2018-09-01	43	43.0	42
4	2018-10-01	36	36.0	37
5	2018-11-01	39	39.0	42

```
percentageDiff.columns=['year_month','controlCustomers','scaledControlNcus','trialCustomers']
percentageDiff
```

	year_month	controlCustomers	scaledControlNcus	trialCustomers
0	2018-06-01	1	1.0	2
1	2018-07-01	51	51.0	50
2	2018-08-01	49	49.0	45
3	2018-09-01	43	43.0	42
4	2018-10-01	36	36.0	37
5	2018-11-01	39	39.0	42
6	2018-12-01	45	45.0	46
7	2019-01-01	36	36.0	36
8	2019-02-01	46	46.0	46
9	2019-03-01	38	38.0	51
10	2019-04-01	36	36.0	45
11	2019-05-01	51	51.0	55
12	2019-06-01	41	41.0	40

```
percentageDiff['%Diff'] = percentageDiff.apply(lambda row: (row['scaledControlNcus']-row['trialCustomers'])/row['scaledControlNcus
percentageDiff
```

	year_month	controlCustomers	scaledControlNcus	trialCustomers	%Diff
0	2018-06-01	1	1.0	2	-1.000000
1	2018-07-01	51	51.0	50	0.019608
2	2018-08-01	49	49.0	45	0.081633
3	2018-09-01	43	43.0	42	0.023256
4	2018-10-01	36	36.0	37	-0.027778
5	2018-11-01	39	39.0	42	-0.076923
6	2018-12-01	45	45.0	46	-0.022222
7	2019-01-01	36	36.0	36	0.000000
8	2019-02-01	46	46.0	46	0.000000
9	2019-03-01	38	38.0	51	-0.342105
10	2019-04-01	36	36.0	45	-0.250000
11	2019-05-01	51	51.0	55	-0.078431
12	2019-06-01	41	41.0	40	0.024390

Get standard deviation

```
stdDev = percentageDiff.loc[percentageDiff['year_month']< '2019-02-01', '%Diff'].std(ddof=8-1)
stdDev
```

```
0.9429551179460401
```

Calculate the t-values for the trial months

```
percentageDiff['t-value'] = percentageDiff.apply(lambda row: (row['%Diff']- 0) / stdDev,axis=1)
percentageDiff
```

	year_month	controlCustomers	scaledControlNcus	trialCustomers	%Diff	t-value
0	2018-06-01	1	1.0	2	-1.000000	-1.060496
1	2018-07-01	51	51.0	50	0.019608	0.020794
2	2018-08-01	49	49.0	45	0.081633	0.086571
3	2018-09-01	43	43.0	42	0.023256	0.024663
4	2018-10-01	36	36.0	37	-0.027778	-0.029458
5	2018-11-01	39	39.0	42	-0.076923	-0.081577
6	2018-12-01	45	45.0	46	-0.022222	-0.023567
7	2019-01-01	36	36.0	36	0.000000	0.000000
8	2019-02-01	46	46.0	46	0.000000	0.000000
9	2019-03-01	38	38.0	51	-0.342105	-0.362801
10	2019-04-01	36	36.0	45	-0.250000	-0.265124

95th & 5th percentile of control store

`measureOverTimeCusts`

year_month	Store_type	nCustomers
2018-06-01	Control	1.000000
2018-06-01	Other stores	3.304147

```
pastNcus_Controls95 = measureOverTimeCusts.loc[measureOverTimeCusts['Store_type']=='Control']
pastNcus_Controls95['nCustomers'] = pastNcus_Controls95.apply(lambda row: row['nCustomers']*(1+stdDev*2),axis=1)
pastNcus_Controls95.iloc[0:13,0] = 'Control 95th % confidence interval'
pastNcus_Controls95.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
```

	year_month	Store_type	nCustomers
0	2018-06-01	Control 95th % confidence interval	2.885910
1	2018-07-01	Control 95th % confidence interval	147.181422
2	2018-08-01	Control 95th % confidence interval	141.409602
3	2018-09-01	Control 95th % confidence interval	124.094140
4	2018-10-01	Control 95th % confidence interval	103.892768
5	2018-11-01	Control 95th % confidence interval	112.550499
6	2018-12-01	Control 95th % confidence interval	129.865961
7	2019-01-01	Control 95th % confidence interval	103.892768
8	2019-02-01	Control 95th % confidence interval	132.751871
9	2019-03-01	Control 95th % confidence interval	109.664589
10	2019-04-01	Control 95th % confidence interval	103.892768
11	2019-05-01	Control 95th % confidence interval	147.181422
12	2019-06-01	Control 95th % confidence interval	118.322320

```
pastNcus_Controls5 = measureOverTimeCusts.loc[measureOverTimeCusts['Store_type']=='Control']
pastNcus_Controls5['nCustomers'] = pastNcus_Controls5.apply(lambda row: row['nCustomers']*(1-stdDev*2),axis=1)
pastNcus_Controls5.iloc[0:13,0] = 'Control 5th % confidence interval'
pastNcus_Controls5.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:
```

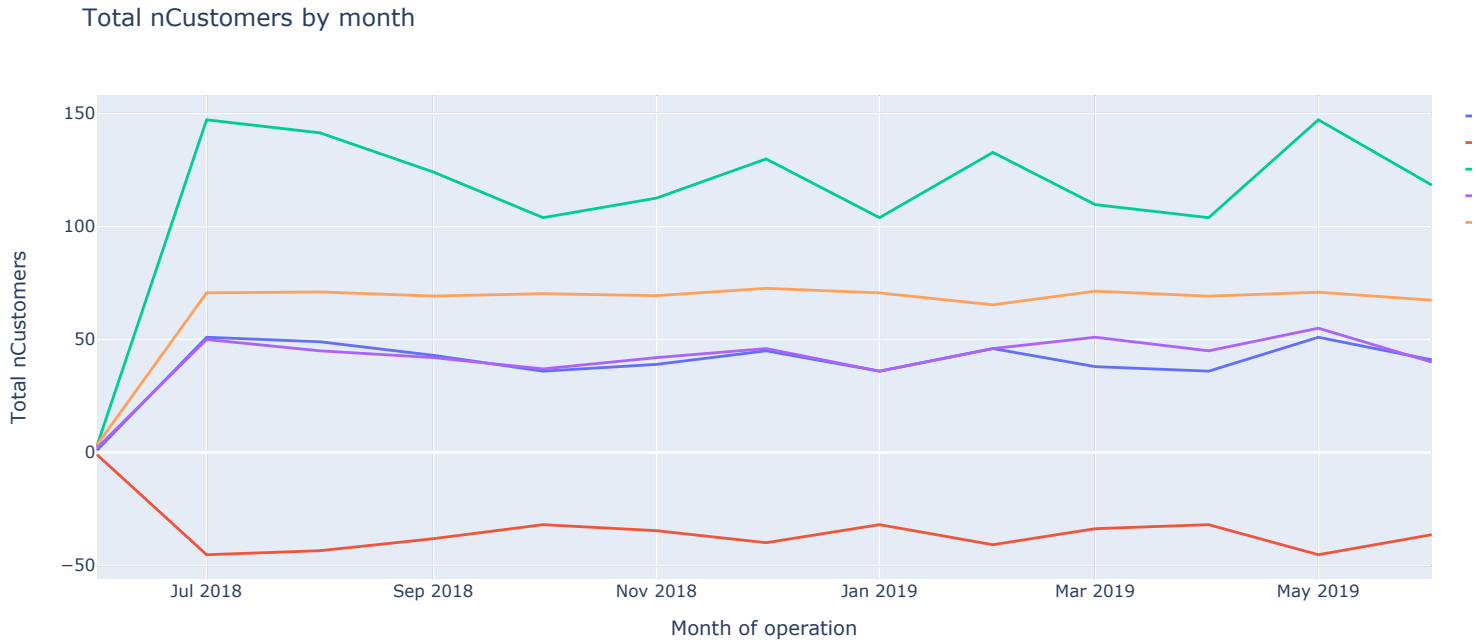
```
trialAssessment = pd.concat([measureOverTimeCusts,pastNcus_Controls5,pastNcus_Controls95])
trialAssessment = trialAssessment.sort_values(by=['year_month'])
trialAssessment = trialAssessment.reset_index()
trialAssessment
```

	year_month	Store_type	nCustomers
0	2018-06-01	Control	1.000000
1	2018-06-01	Control 5th % confidence interval	-0.885910
2	2018-06-01	Control 95th % confidence interval	2.885910
3	2018-06-01	Trail	2.000000
4	2018-06-01	Other stores	3.304147
...	...	...	...
60	2019-06-01	Trail	40.000000
61	2019-06-01	Other stores	67.396947
62	2019-06-01	Control	41.000000
63	2019-06-01	Control 5th % confidence interval	-36.322320
64	2019-06-01	Control 95th % confidence interval	118.322320

65 rows × 3 columns

Visualization Trial

```
px.line(data_frame=trialAssessment, x='year_month', y='nCustomers', color='Store_type', title='Total nCustomers by month',labels=
```



Trial store 86

Select control store

Get correlation

```
preTrialMeasures
```

	year_month	STORE_NBR	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	2018-06-01	1	4.8	2	1.000000	1.000000	2.400000
1	2018-07-01	1	220.2	52	1.057692	1.181818	3.387692
2	2018-08-01	1	171.8	41	1.000000	1.292683	3.241509
3	2018-09-01	1	278.2	59	1.050847	1.209677	3.709333
4	2018-10-01	1	186.4	44	1.022727	1.266667	3.270175
...	...	...	...	...	...	...	...
2062	2018-09-01	272	295.9	31	1.129032	1.971429	4.288406
2063	2018-10-01	272	438.0	45	1.155556	1.942308	4.336634
2064	2018-11-01	272	384.0	42	1.095238	1.934783	4.314607
2065	2018-12-01	272	388.7	45	1.000000	1.888889	4.572941
2066	2019-01-01	272	423.0	46	1.086957	1.920000	4.406250

2067 rows × 7 columns

```
corr_sales_86 = calculateCorrelation(inputTable=preTrialMeasures, metricCol='TOT_SALES',storeComparison='86')
corr_sales_86
```

	Store1	Store2	corr_measure
0	86	1	0.906054
1	86	2	0.904013
2	86	3	0.953057
3	86	4	0.944243
4	86	5	0.972135
...	...	...	...
263	86	268	0.781063
264	86	269	0.988160
265	86	270	0.900343
266	86	271	0.971290
267	86	272	0.931813

268 rows × 3 columns

```
corr_cust_86 = calculateCorrelation(inputTable=preTrialMeasures, metricCol='nCustomers',storeComparison='86')
corr_cust_86
```

	Store1	Store2	corr_measure
0	86	1	0.933287
1	86	2	0.956773
2	86	3	0.972706
3	86	4	0.961919
4	86	5	0.972371
...	...	...	...
263	86	268	0.898179
264	86	269	0.987938
265	86	270	0.920857
266	86	271	0.981490
267	86	272	0.911463

268 rows × 3 columns

Get magnitude and standardise

```
mag_sales_86 = finalDistTable(inputTable=preTrialMeasures, metricCol='TOT_SALES',storeComparison='86')
mag_sales_86
```

	Store1	Store2	mag_measure
0	86	1	0.206684
1	86	2	0.195477
2	86	3	0.681000
3	86	4	0.540193
4	86	5	0.868704
...	...	...	...
263	86	268	0.235064
264	86	269	0.852256
265	86	270	0.734966
266	86	271	0.885476

```
mag_cust_86 = finalDistTable(inputTable=preTrialMeasures, metricCol='nCustomers',storeComparison='86')
mag_cust_86
```

	Store1	Store2	mag_measure
0	86	1	0.444200
1	86	2	0.388768
2	86	3	0.814789
3	86	4	0.793052
4	86	5	0.873234
...	...	...	...
263	86	268	0.406794
264	86	269	0.883173
265	86	270	0.775641
266	86	271	0.919704
267	86	272	0.448918

268 rows × 3 columns

Get scored sales

```
score_86_nSales = corr_sales_86.merge(mag_sales_86, on=['Store1','Store2'])
score_86_nSales['scoreNSales'] = score_86_nSales.apply(lambda row: row['corr_measure']*0.5 + row['mag_measure']*0.5, axis=1)
score_86_nSales = score_86_nSales.loc[:,['Store1','Store2', 'scoreNSales']]
score_86_nSales
```

	Store1	Store2	scoreNSales
0	86	1	0.556369
1	86	2	0.549745
2	86	3	0.817028
3	86	4	0.742218
4	86	5	0.920419
...	...	...	...
263	86	268	0.508063
264	86	269	0.920208
265	86	270	0.817654
266	86	271	0.928383
267	86	272	0.699771

268 rows × 3 columns

Get scored nCustomers

```
score_86_nCust = corr_cust_86.merge(mag_cust_86, on=['Store1','Store2'])
score_86_nCust['scoreNcust'] = score_86_nCust.apply(lambda row: row['corr_measure']*0.5 + row['mag_measure']*0.5, axis=1)
score_86_nCust = score_86_nCust.loc[:,['Store1','Store2', 'scoreNcust']]
score_86_nCust
```

	Store1	Store2	scoreNcust
0	86	1	0.688744
1	86	2	0.672771
2	86	3	0.893748
3	86	4	0.877485
4	86	5	0.922803
...	...	...	...
263	86	268	0.652487
264	86	269	0.935555
265	86	270	0.848249
266	86	271	0.950597
267	86	272	0.680190

Combine scored table

```
score_Control_86 = score_86_nSales.merge(score_86_nCust, on=['Store1','Store2'])
score_Control_86
```

	Store1	Store2	scoreNSales	scoreNcust
0	86	1	0.556369	0.688744
1	86	2	0.549745	0.672771
2	86	3	0.817028	0.893748
3	86	4	0.742218	0.877485
4	86	5	0.920419	0.922803
...	...	...	...	...
263	86	268	0.508063	0.652487
264	86	269	0.920208	0.935555
265	86	270	0.817654	0.848249
266	86	271	0.928383	0.950597
267	86	272	0.699771	0.680190

268 rows × 4 columns

```
score_Control_86['finalControlScore'] = score_Control_86.apply(lambda row: row['scoreNSales']*0.5 + row['scoreNcust']*0.5, axis=1)
score_Control_86
```

	Store1	Store2	scoreNSales	scoreNcust	finalControlScore
0	86	1	0.556369	0.688744	0.622556
1	86	2	0.549745	0.672771	0.611258
2	86	3	0.817028	0.893748	0.855388
3	86	4	0.742218	0.877485	0.809852
4	86	5	0.920419	0.922803	0.921611
...	...	...	...	...	...
263	86	268	0.508063	0.652487	0.580275
264	86	269	0.920208	0.935555	0.927882
265	86	270	0.817654	0.848249	0.832952
266	86	271	0.928383	0.950597	0.939490
267	86	272	0.699771	0.680190	0.689981

268 rows × 5 columns

Get control store

```
final_control_store_86 = score_Control_86['finalControlScore'].max()
final_control_store_86
```

```
0.9638823649111354
```

```
score_Control_86.loc[score_Control_86['finalControlScore']==0.9638823649111354]
```

	Store1	Store2	scoreNSales	scoreNcust	finalControlScore
104	86	109	0.96593	0.961834	0.963882

```
score_Control_86.loc[score_Control_86['Store2']==155]
```

	Store1	Store2	scoreNSales	scoreNcust	finalControlScore
150	86	155	0.937389	0.952506	0.944948

Visualization the control store

```
ime86 = measureOverTime
ime86['Store_type'] = measureOverTime86.apply(lambda row: 'Trail' if row['STORE_NBR']==86 else ('Control' if row['STORE_NBR']==10
ime86
```

	STORE_NBR	year_month	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	Store_type
0	1	2018-06-01	4.8	2	1.000000	1.000000	2.400000	Other stores
1	1	2018-07-01	220.2	52	1.057692	1.181818	3.387692	Other stores
2	1	2018-08-01	171.8	41	1.000000	1.292683	3.241509	Other stores
3	1	2018-09-01	278.2	59	1.050847	1.209677	3.709333	Other stores
4	1	2018-10-01	186.4	44	1.022727	1.266667	3.270175	Other stores
...	...	...	...	...	...	...	...	...
3382	272	2019-02-01	395.5	45	1.066667	1.895833	4.346154	Other stores
3383	272	2019-03-01	478.5	52	1.115385	1.913793	4.310811	Other stores
3384	272	2019-04-01	415.5	50	1.040000	1.865385	4.283505	Other stores
3385	272	2019-05-01	322.8	36	1.138889	1.780488	4.421918	Other stores
3386	272	2019-06-01	297.3	32	1.093750	1.885714	4.504545	Other stores

3387 rows × 8 columns

```
measureOverTimeSales86 = measureOverTime86.groupby(['year_month','Store_type'])['totSales'].mean().reset_index()
measureOverTimeSales86
```

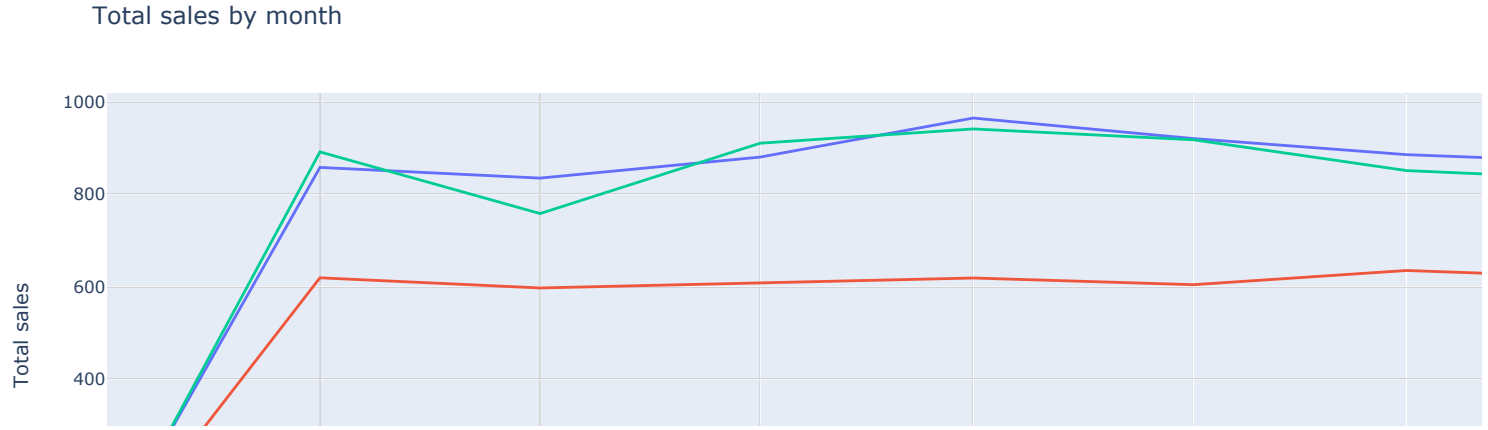
	year_month	Store_type	totSales
0	2018-06-01	Control	45.800000
1	2018-06-01	Other stores	24.371429
2	2018-06-01	Trail	37.800000
3	2018-07-01	Control	858.000000
4	2018-07-01	Other stores	619.319697
5	2018-07-01	Trail	891.600000
6	2018-08-01	Control	834.900000
7	2018-08-01	Other stores	597.364885
8	2018-08-01	Trail	758.250000
9	2018-09-01	Control	880.400000
10	2018-09-01	Other stores	607.525287
11	2018-09-01	Trail	910.400000
12	2018-10-01	Control	964.800000
13	2018-10-01	Other stores	618.888973

```
measureOverTimeSales86.set_index('year_month',inplace=True)


pastSales86 = measureOverTimeSales86.loc['2018-06-01':'2019-01-01'].reset_index()
pastSales86
```

	year_month	Store_type	totSales
0	2018-06-01	Control	45.800000
1	2018-06-01	Other stores	24.371429
2	2018-06-01	Trail	37.800000
3	2018-07-01	Control	858.000000
4	2018-07-01	Other stores	619.319697
5	2018-07-01	Trail	891.600000
6	2018-08-01	Control	834.900000
7	2018-08-01	Other stores	597.364885
8	2018-08-01	Trail	758.250000
9	2018-09-01	Control	880.400000
10	2018-09-01	Other stores	607.525287
11	2018-09-01	Trail	910.400000
12	2018-10-01	Control	964.800000
13	2018-10-01	Other stores	618.888973
14	2018-10-01	Trail	941.200000
15	2018-11-01	Control	920.400000
16	2018-11-01	Other stores	604.440840
17	2018-11-01	Trail	917.800000
18	2018-12-01	Control	885.600000
19	2018-12-01	Other stores	635.134483
20	2018-12-01	Trail	851.200000
21	2019-01-01	Control	867.600000
22	2019-01-01	Other stores	618.401533
23	2019-01-01	Trail	830.200000

```
ɛ.line(data_frame=pastSales86, x='year_month', y='totSales', color='Store_type', title='Total sales by month',labels={'year_month'
```



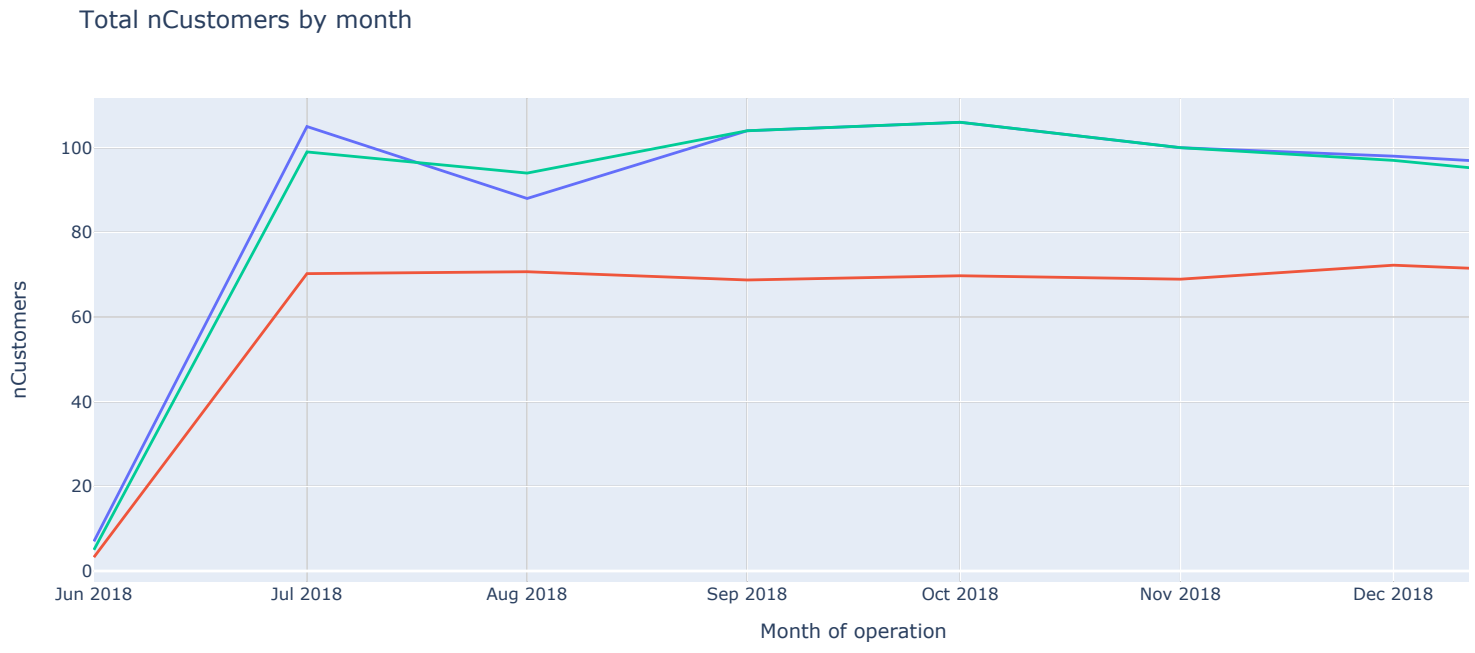

```
measureOverTimeCust86 = measureOverTime86.groupby(['year_month','Store_type'])['nCustomers'].mean().reset_index()
measureOverTimeCust86.set_index('year_month',inplace=True)
measureOverTimeCust86
```

year_month	Store_type	nCustomers

```
pastCust86 = measureOverTimeCust86.loc['2018-06-01':'2019-01-01'].reset_index()
pastCust86
```

	year_month	Store_type	nCustomers
0	2018-06-01	Control	7.000000
1	2018-06-01	Other stores	3.262673
2	2018-06-01	Trail	5.000000
3	2018-07-01	Control	105.000000
4	2018-07-01	Other stores	70.250000
5	2018-07-01	Trail	99.000000
6	2018-08-01	Control	88.000000
7	2018-08-01	Other stores	70.698473
8	2018-08-01	Trail	94.000000
9	2018-09-01	Control	104.000000
10	2018-09-01	Other stores	68.750958
11	2018-09-01	Trail	104.000000
12	2018-10-01	Control	106.000000
13	2018-10-01	Other stores	69.745247
14	2018-10-01	Trail	106.000000
15	2018-11-01	Control	100.000000
16	2018-11-01	Other stores	68.938931
17	2018-11-01	Trail	100.000000
18	2018-12-01	Control	98.000000
19	2018-12-01	Other stores	72.241379
20	2018-12-01	Trail	97.000000
21	2019-01-01	Control	95.000000
22	2019-01-01	Other stores	70.172414
23	2019-01-01	Trail	92.000000

```
x='year_month', y='nCustomers', color='Store_type', title='Total nCustomers by month',labels={'year_month':'Month of operation',
```



Assessment of trial period

Calculate for totSales

Scale sales

```
scalingFactorForControlSales86 = preTrialMeasures.loc[preTrialMeasures['STORE_NBR']==86, 'TOT_SALES'].sum() / preTrialMeasures.loc
scalingFactorForControlSales86
```

```
0.9809748302037556
```

Apply scaling factor

```
scaledControlSales86 = measureOverTimeSales86.loc[measureOverTimeSales86['Store_type']=='Control','totSales'].reset_index()
scaledControlSales86
```

	year_month	totSales
0	2018-06-01	45.8
1	2018-07-01	858.0
2	2018-08-01	834.9
3	2018-09-01	880.4
4	2018-10-01	964.8
5	2018-11-01	920.4
6	2018-12-01	885.6
7	2019-01-01	867.6
8	2019-02-01	864.6
9	2019-03-01	1023.8
10	2019-04-01	736.0
11	2019-05-01	737.6
12	2019-06-01	779.6

```
scaledControlSales86['scaledControlSales'] = scaledControlSales86.apply(lambda row: row['totSales']*scalingFactorForControlSales8
scaledControlSales86
```

	year_month	totSales	scaledControlSales
0	2018-06-01	45.8	44.928647
1	2018-07-01	858.0	841.676404
2	2018-08-01	834.9	819.015886
3	2018-09-01	880.4	863.650241
4	2018-10-01	964.8	946.444516
5	2018-11-01	920.4	902.889234
6	2018-12-01	885.6	868.751310
7	2019-01-01	867.6	851.093763
8	2019-02-01	864.6	848.150838
9	2019-03-01	1023.8	1004.322031
10	2019-04-01	736.0	721.997475
11	2019-05-01	737.6	723.567035
12	2019-06-01	779.6	764.767978

```
TrailStoreSales86 = measureOverTimeSales86.loc[measureOverTimeSales86['Store_type']=='Trail',['totSales']]
TrailStoreSales86
```

year_month	totSales
2018-06-01	37.80
2018-07-01	891.60
2018-08-01	758.25
2018-09-01	910.40
2018-10-01	941.20
2018-11-01	917.80

```
TrailStoreSales86.columns = ['trailSales']
TrailStoreSales86
```

year_month	trailSales
2018-06-01	37.80
2018-07-01	891.60
2018-08-01	758.25
2018-09-01	910.40
2018-10-01	941.20
2018-11-01	917.80
2018-12-01	851.20
2019-01-01	830.20
2019-02-01	939.20
2019-03-01	1002.40
2019-04-01	869.60
2019-05-01	868.50
2019-06-01	817.20

%Diff between scaled control & trail store

```
percentageDiff86 = scaledControlSales86.merge(TrailStoreSales86, on='year_month')
percentageDiff86
```

	year_month	totSales	scaledControlSales	trailSales
0	2018-06-01	45.8	44.928647	37.80
1	2018-07-01	858.0	841.676404	891.60
2	2018-08-01	834.9	819.015886	758.25
3	2018-09-01	880.4	863.650241	910.40
4	2018-10-01	964.8	946.444516	941.20
5	2018-11-01	920.4	902.889234	917.80
6	2018-12-01	885.6	868.751310	851.20
7	2019-01-01	867.6	851.093763	830.20
8	2019-02-01	864.6	848.150838	939.20
9	2019-03-01	1023.8	1004.322031	1002.40
10	2019-04-01	736.0	721.997475	869.60
11	2019-05-01	737.6	723.567035	868.50
12	2019-06-01	779.6	764.767978	817.20

```
percentageDiff86['percentDiff'] = percentageDiff86.apply(lambda row: (row['scaledControlSales']-row['trailSales'])/row['scaledCont
percentageDiff86
```

	year_month	totSales	scaledControlSales	trailSales	percentDiff
0	2018-06-01	45.8	44.928647	37.80	0.158666
1	2018-07-01	858.0	841.676404	891.60	-0.059314
2	2018-08-01	834.9	819.015886	758.25	0.074194
3	2018-09-01	880.4	863.650241	910.40	-0.054130
4	2018-10-01	964.8	946.444516	941.20	0.005541
5	2018-11-01	920.4	902.889234	917.80	-0.016515
6	2018-12-01	885.6	868.751310	851.20	0.020203
7	2019-01-01	867.6	851.093763	830.20	0.024549

Get standard deviation

9	2019-03-01	1023.8	1004.322031	1002.40	0.001914

```
stdDev = percentageDiff86.loc[percentageDiff86['year_month']< '2019-02-01', 'percentDiff'].std(ddof=8-1)
stdDev
```

```
0.18843734625185254
```

Calculate t-values

```
percentageDiff86['t-value'] = percentageDiff86.apply(lambda row: (row['percentDiff']- 0) / stdDev,axis=1)
percentageDiff86
```

	year_month	totSales	scaledControlSales	trailSales	percentDiff	t-value
0	2018-06-01	45.8	44.928647	37.80	0.158666	0.842009
1	2018-07-01	858.0	841.676404	891.60	-0.059314	-0.314770
2	2018-08-01	834.9	819.015886	758.25	0.074194	0.393732
3	2018-09-01	880.4	863.650241	910.40	-0.054130	-0.287260
4	2018-10-01	964.8	946.444516	941.20	0.005541	0.029406
5	2018-11-01	920.4	902.889234	917.80	-0.016515	-0.087639
6	2018-12-01	885.6	868.751310	851.20	0.020203	0.107213
7	2019-01-01	867.6	851.093763	830.20	0.024549	0.130278
8	2019-02-01	864.6	848.150838	939.20	-0.107350	-0.569686
9	2019-03-01	1023.8	1004.322031	1002.40	0.001914	0.010156
10	2019-04-01	736.0	721.997475	869.60	-0.204436	-1.084904
11	2019-05-01	737.6	723.567035	868.50	-0.200303	-1.062971
12	2019-06-01	779.6	764.767978	817.20	-0.068559	-0.363831

95th & 5th percentile of control store

```
pastSales_86_Controls95 = measureOverTimeSales86.loc[measureOverTimeSales86['Store_type']=='Control']
pastSales_86_Controls95['totSales'] = pastSales_86_Controls95.apply(lambda row: row['totSales']*(1+stdDev*2),axis=1)
pastSales_86_Controls95.iloc[0:13,0] = 'Control 95th % confidence interval'
pastSales_86_Controls95.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
```

year_month	Store_type	totSales

```
pastSales_86_Controls5 = measureOverTimeSales86.loc[measureOverTimeSales86['Store_type']=='Control']
pastSales_86_Controls5['totSales'] = pastSales_86_Controls5.apply(lambda row: row['totSales']*(1-stdDev*2),axis=1)
pastSales_86_Controls5.iloc[0:13,0] = 'Control 5th % confidence interval'
pastSales_86_Controls5.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
```

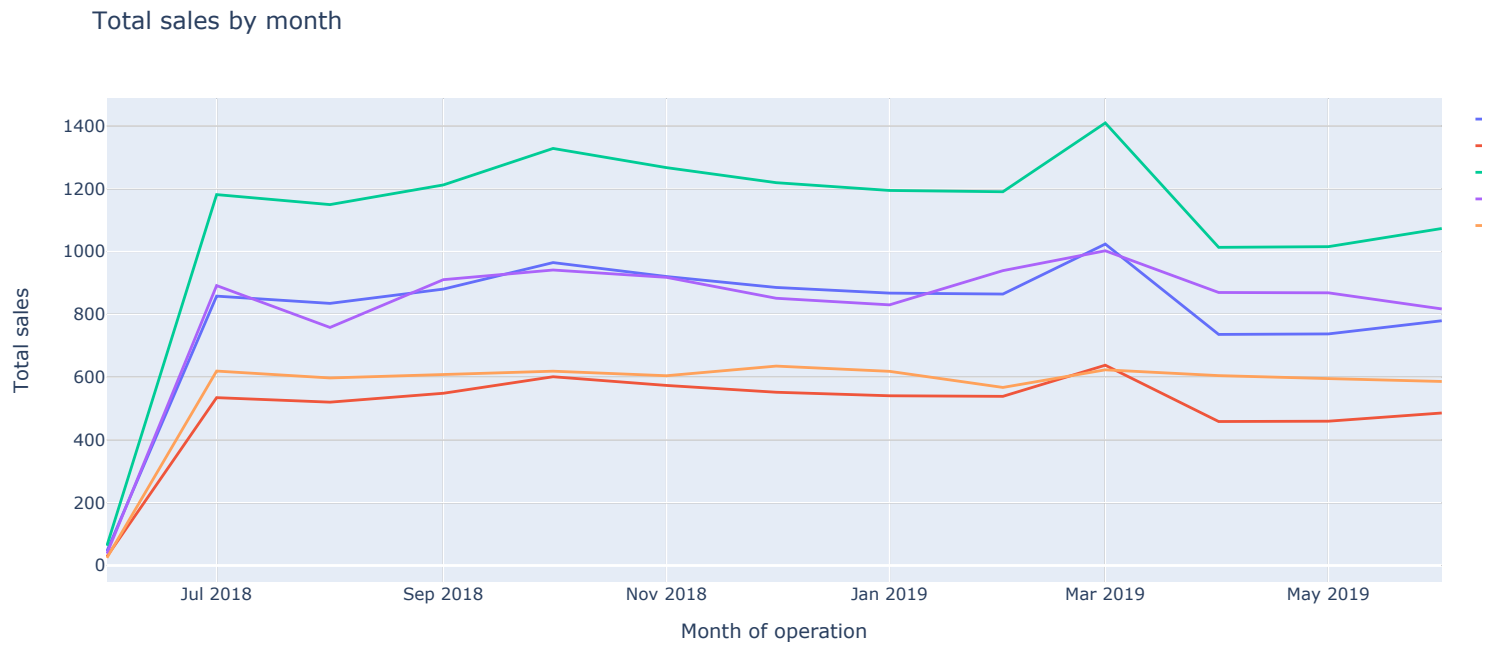
	year_month	Store_type	totSales
0	2018-06-01	Control 5th % confidence interval	28.539139
1	2018-07-01	Control 5th % confidence interval	534.641514
2	2018-08-01	Control 5th % confidence interval	520.247319
3	2018-09-01	Control 5th % confidence interval	548.599521
4	2018-10-01	Control 5th % confidence interval	601.191297
5	2018-11-01	Control 5th % confidence interval	573.524533
6	2018-12-01	Control 5th % confidence interval	551.839772
7	2019-01-01	Control 5th % confidence interval	540.623517
8	2019-02-01	Control 5th % confidence interval	538.754141
9	2019-03-01	Control 5th % confidence interval	637.955690
10	2019-04-01	Control 5th % confidence interval	458.620226
11	2019-05-01	Control 5th % confidence interval	459.617227
12	2019-06-01	Control 5th % confidence interval	485.788490

```
trialAssessment_sales_86 = pd.concat([measureOverTimeSales86,pastSales_86_Controls5,pastSales_86_Controls95])
trialAssessment_sales_86 = trialAssessment_sales_86.sort_values(by=['year_month'])
trialAssessment_sales_86 = trialAssessment_sales_86.reset_index()
trialAssessment_sales_86
```

	year_month	Store_type	totSales

Visualization Trial

```
nt_sales_86, x='year_month', y='totSales', color='Store_type', title='Total sales by month',labels={'year_month':'Month of operat
```



Calculate for nCustomers

Scale nCustomers

```
scalingFactorForControlNcust86 = preTrialMeasures.loc[preTrialMeasures['STORE_NBR']==86, 'nCustomers'].sum() / preTrialMeasures.lo
scalingFactorForControlNcust86
```

```
0.9914651493598862
```

Apply scaling factor

```
scaledControlNcust86 = measureOverTimeCust86.loc[measureOverTimeCust86['Store_type']=='Control','nCustomers'].reset_index()
scaledControlNcust86
```

	year_month	nCustomers
0	2018-06-01	7.0
1	2018-07-01	105.0
2	2018-08-01	88.0
3	2018-09-01	104.0
4	2018-10-01	106.0
5	2018-11-01	100.0
6	2018-12-01	98.0
7	2019-01-01	95.0
8	2019-02-01	94.0
9	2019-03-01	112.0
10	2019-04-01	81.0
11	2019-05-01	90.0
12	2019-06-01	88.0

```
scaledControlNcust86['scaledControlNcust'] = scaledControlNcust86.apply(lambda row: row['nCustomers']*scalingFactorForControlNcust
# scaledControlNcust86 = scaledControlNcust86.drop(['scaledControlSales'],axis=1)
scaledControlNcust86
```

	year_month	nCustomers	scaledControlNcust
0	2018-06-01	7.0	6.940256
1	2018-07-01	105.0	104.103841
2	2018-08-01	88.0	87.248933
3	2018-09-01	104.0	103.112376
4	2018-10-01	106.0	105.095306
5	2018-11-01	100.0	99.146515
6	2018-12-01	98.0	97.163585
7	2019-01-01	95.0	94.189189
8	2019-02-01	94.0	93.197724
9	2019-03-01	112.0	111.044097
10	2019-04-01	81.0	80.308677
11	2019-05-01	90.0	89.231863
12	2019-06-01	88.0	87.248933

```
TrailStoreNcust86 = measureOverTimeCust86.loc[measureOverTimeCust86['Store_type']=='Trail',['nCustomers']]
TrailStoreNcust86
```

year_month	nCustomers
2018-06-01	5.0
2018-07-01	99.0
2018-08-01	94.0
2018-09-01	104.0
2018-10-01	106.0
2018-11-01	100.0
2018-12-01	97.0
2019-01-01	92.0
2019-02-01	108.0
2019-03-01	112.0
2019-04-01	106.0
2019-05-01	102.0
2019-06-01	96.0

```
TrailStoreNcust86.columns = ['trailNcust']
TrailStoreNcust86
```

year_month	trailNcust
2018-06-01	5.0
2018-07-01	99.0
2018-08-01	94.0
2018-09-01	104.0
2018-10-01	106.0
2018-11-01	100.0
2018-12-01	97.0
2019-01-01	92.0
2019-02-01	108.0
2019-03-01	112.0
2019-04-01	106.0
2019-05-01	102.0
2019-06-01	96.0

%Diff between scaled control & trail store

```
percentageDiff86 = scaledControlNcust86.merge(TrailStoreNcust86, on='year_month')
```

```
percentageDiff86
```

	year_month	nCustomers	scaledControlNcust	trailNcust
0	2018-06-01	7.0	6.940256	5.0
1	2018-07-01	105.0	104.103841	99.0
2	2018-08-01	88.0	87.248933	94.0
3	2018-09-01	104.0	103.112376	104.0
4	2018-10-01	106.0	105.095306	106.0
5	2018-11-01	100.0	99.146515	100.0
6	2018-12-01	98.0	97.163585	97.0
7	2019-01-01	95.0	94.189189	92.0
8	2019-02-01	94.0	93.197724	108.0
9	2019-03-01	112.0	111.044097	112.0
10	2019-04-01	81.0	80.308677	106.0
11	2019-05-01	90.0	89.231863	102.0
12	2019-06-01	88.0	87.248933	96.0

```
percentageDiff86['percentDiff'] = percentageDiff86.apply(lambda row: (row['scaledControlNcust']-row['trailNcust'])/row['scaledCont
percentageDiff86
```

	year_month	nCustomers	scaledControlNcust	trailNcust	percentDiff
0	2018-06-01	7.0	6.940256	5.0	0.279565
1	2018-07-01	105.0	104.103841	99.0	0.049026
2	2018-08-01	88.0	87.248933	94.0	-0.077377
3	2018-09-01	104.0	103.112376	104.0	-0.008608
4	2018-10-01	106.0	105.095306	106.0	-0.008608
5	2018-11-01	100.0	99.146515	100.0	-0.008608
6	2018-12-01	98.0	97.163585	97.0	0.001684
7	2019-01-01	95.0	94.189189	92.0	0.023242
8	2019-02-01	94.0	93.197724	108.0	-0.158827
9	2019-03-01	112.0	111.044097	112.0	-0.008608
10	2019-04-01	81.0	80.308677	106.0	-0.319907
11	2019-05-01	90.0	89.231863	102.0	-0.143089
12	2019-06-01	88.0	87.248933	96.0	-0.100300

Get standard deviation

```
stdDev = percentageDiff86.loc[percentageDiff86['year_month']< '2019-02-01', 'percentDiff'].std(ddof=8-1)
stdDev
```

```
0.28192332231660333
```

Calculate t-values

```
percentageDiff86['t-value'] = percentageDiff86.apply(lambda row: (row['percentDiff']- 0) / stdDev,axis=1)
percentageDiff86
```

	year_month	nCustomers	scaledControlNcust	trailNcust	percentDiff	t-value
0	2018-06-01	7.0	6.940256	5.0	0.279565	0.991637
1	2018-07-01	105.0	104.103841	99.0	0.049026	0.173900
2	2018-08-01	88.0	87.248933	94.0	-0.077377	-0.274461

95th & 5th percentile of control store

```
pastCust_86_Controls95 = measureOverTimeCust86.loc[measureOverTimeCust86['Store_type']=='Control']
pastCust_86_Controls95['nCustomers'] = pastCust_86_Controls95.apply(lambda row: row['nCustomers']*(1+stdDev*2),axis=1)
pastCust_86_Controls95.iloc[0:13,0] = 'Control 95th % confidence interval'
pastCust_86_Controls95.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
```

	year_month	Store_type	nCustomers
0	2018-06-01	Control 95th % confidence interval	10.946927
1	2018-07-01	Control 95th % confidence interval	164.203898
2	2018-08-01	Control 95th % confidence interval	137.618505
3	2018-09-01	Control 95th % confidence interval	162.640051
4	2018-10-01	Control 95th % confidence interval	165.767744
5	2018-11-01	Control 95th % confidence interval	156.384664
6	2018-12-01	Control 95th % confidence interval	153.256971
7	2019-01-01	Control 95th % confidence interval	148.565431
8	2019-02-01	Control 95th % confidence interval	147.001585
9	2019-03-01	Control 95th % confidence interval	175.150824
10	2019-04-01	Control 95th % confidence interval	126.671578
11	2019-05-01	Control 95th % confidence interval	140.746198
12	2019-06-01	Control 95th % confidence interval	137.618505

```
pastCust_86_Controls5 = measureOverTimeCust86.loc[measureOverTimeCust86['Store_type']=='Control']
pastCust_86_Controls5['nCustomers'] = pastCust_86_Controls5.apply(lambda row: row['nCustomers']*(1-stdDev*2),axis=1)
pastCust_86_Controls5.iloc[0:13,0] = 'Control 5th % confidence interval'
pastCust_86_Controls5.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
```

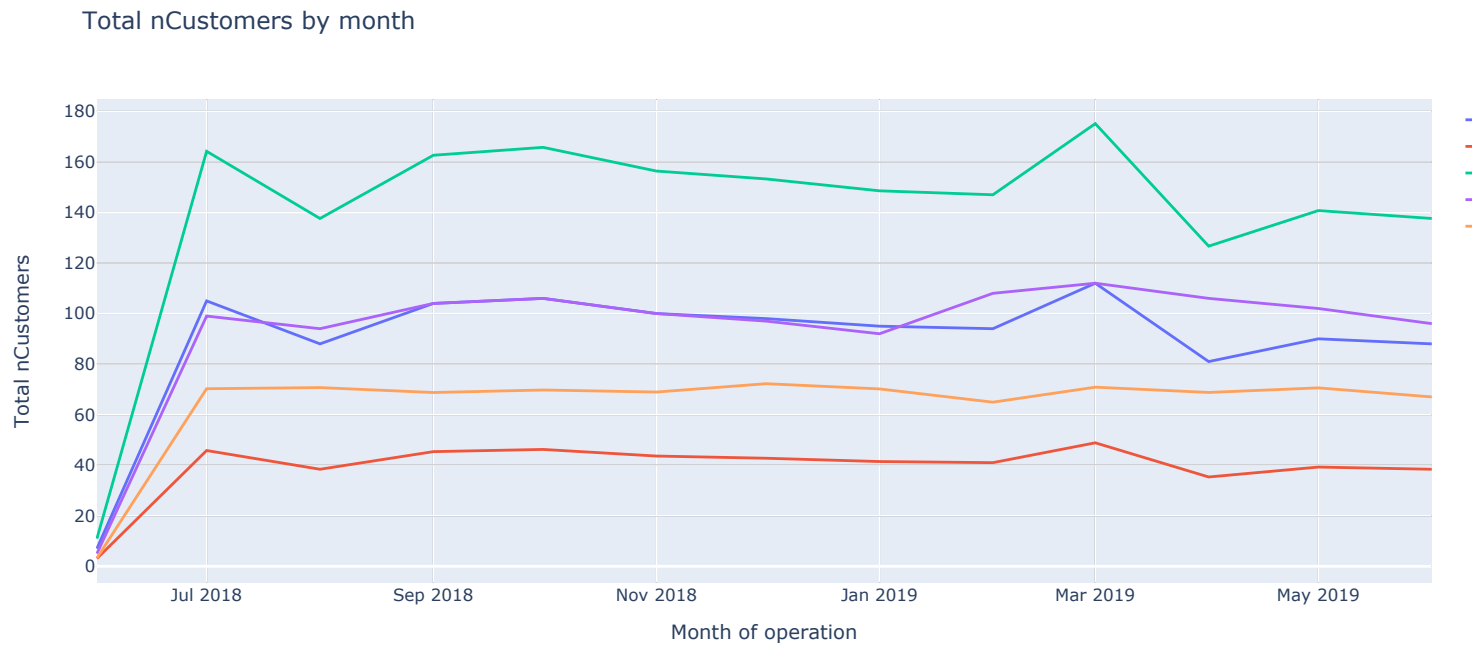
```
trialAssessment_cust_86 = pd.concat([measureOverTimeCust86,pastCust_86_Controls5,pastCust_86_Controls95])
trialAssessment_cust_86 = trialAssessment_cust_86.sort_values(by=['year_month'])
trialAssessment_cust_86 = trialAssessment_cust_86.reset_index()
trialAssessment_cust_86
```

	year_month	Store_type	nCustomers
0	2018-06-01	Control	7.000000
1	2018-06-01	Control 5th % confidence interval	3.053073
2	2018-06-01	Control 95th % confidence interval	10.946927
3	2018-06-01	Trail	5.000000
4	2018-06-01	Other stores	3.262673
...	...	...	...
60	2019-06-01	Trail	96.000000
61	2019-06-01	Other stores	67.003817
62	2019-06-01	Control	88.000000
63	2019-06-01	Control 5th % confidence interval	38.381495
64	2019-06-01	Control 95th % confidence interval	137.618505

65 rows × 3 columns

Visualization Trial

```
px.line(data_frame=trialAssessment_cust_86, x='year_month', y='nCustomers', color='Store_type', title='Total nCustomers by month',
```



Trial store 88

Select control store

Get correlation

```
preTrialMeasures
```

	year_month	STORE_NBR	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	2018-06-01	1	4.8	2	1.000000	1.000000	2.400000
1	2018-07-01	1	220.2	52	1.057692	1.181818	3.387692
2	2018-08-01	1	171.8	41	1.000000	1.292683	3.241509
3	2018-09-01	1	278.2	59	1.050847	1.209677	3.709333
4	2018-10-01	1	186.4	44	1.022727	1.266667	3.270175
...	...	...	...	...	...	...	...
2062	2018-09-01	272	295.9	31	1.129032	1.971429	4.288406
2063	2018-10-01	272	438.0	45	1.155556	1.942308	4.336634
2064	2018-11-01	272	384.0	42	1.095238	1.934783	4.314607
2065	2018-12-01	272	388.7	45	1.000000	1.888889	4.572941
2066	2019-01-01	272	423.0	46	1.086957	1.920000	4.406250

2067 rows × 7 columns

```
corr_sales_88 = calculateCorrelation(inputTable=preTrialMeasures, metricCol='TOT_SALES', storeComparison=88)
corr_sales_88
```

	Store1	Store2	corr_measure
0	88.0	1.0	0.906059
1	88.0	2.0	0.941415
2	88.0	3.0	0.960514
3	88.0	4.0	0.953270
4	88.0	5.0	0.984270
...	...	...	...
263	88.0	268.0	0.832364
264	88.0	269.0	0.963062
265	88.0	270.0	0.939474
266	88.0	271.0	0.954349
267	88.0	272.0	0.915255

268 rows × 3 columns

```
corr_cust_88 = calculateCorrelation(inputTable=preTrialMeasures, metricCol='nCustomers', storeComparison=88)
corr_cust_88
```

	Store1	Store2	corr_measure
0	88.0	1.0	0.908413
1	88.0	2.0	0.955055
2	88.0	3.0	0.988394
3	88.0	4.0	0.972464
4	88.0	5.0	0.987158
...	...	...	...
263	88.0	268.0	0.921433
264	88.0	269.0	0.986109
265	88.0	270.0	0.959026
266	88.0	271.0	0.974388
267	88.0	272.0	0.932375

268 rows × 3 columns

Get magnitude and standardise

```
mag_sales_88 = finalDistTable(inputTable=preTrialMeasures, metricCol='TOT_SALES', storeComparison=88)
mag sales 88
```

	Store1	Store2	mag_measure
0	88	1	0.136529
1	88	2	0.132578
2	88	3	0.728723
3	88	4	0.913857
4	88	5	0.578280
...	...	...	...
263	88	268	0.154230
264	88	269	0.675669
265	88	270	0.676782
266	88	271	0.651379
267	88	272	0.317338

268 rows × 3 columns

```
mag_cust_88 = finalDistTable(inputTable=preTrialMeasures, metricCol='nCustomers', storeComparison=88)
mag_cust_88
```

	Store1	Store2	mag_measure
0	88	1	0.337218
1	88	2	0.292381
2	88	3	0.748751
3	88	4	0.921175
4	88	5	0.680101
...	...	...	...
263	88	268	0.300456
264	88	269	0.791654
265	88	270	0.760114
266	88	271	0.774675
267	88	272	0.344291

268 rows × 3 columns

Get score sales & nCustomers

```
score_88_nSales = corr_sales_88.merge(mag_sales_88, on=['Store1','Store2'])
score_88_nSales['scoreNSales'] = score_88_nSales.apply(lambda row: row['corr_measure']*0.5 + row['mag_measure']*0.5, axis=1)
score_88_nSales = score_88_nSales.loc[:,['Store1','Store2', 'scoreNSales']]
score_88_nSales
```

	Store1	Store2	scoreNSales
0	88.0	1.0	0.521294
1	88.0	2.0	0.536997
2	88.0	3.0	0.844619
3	88.0	4.0	0.933564
4	88.0	5.0	0.781275
...	...	...	...
263	88.0	268.0	0.493297
264	88.0	269.0	0.819365
265	88.0	270.0	0.808128
266	88.0	271.0	0.802864
267	88.0	272.0	0.616297

268 rows × 3 columns

```
score_88_nCust = corr_cust_88.merge(mag_cust_88, on=['Store1','Store2'])
score_88_nCust['scoreNcust'] = score_88_nCust.apply(lambda row: row['corr_measure']*0.5 + row['mag_measure']*0.5, axis=1)
score_88_nCust = score_88_nCust.loc[:,['Store1','Store2', 'scoreNcust']]
score_88_nCust
```

	Store1	Store2	scoreNcust
0	88.0	1.0	0.622816
1	88.0	2.0	0.623718
2	88.0	3.0	0.868572
3	88.0	4.0	0.946820
4	88.0	5.0	0.833630
...	...	...	...
263	88.0	268.0	0.610945
264	88.0	269.0	0.888881
265	88.0	270.0	0.859570
266	88.0	271.0	0.874531
267	88.0	272.0	0.638333

268 rows × 3 columns

Combine score table

```
score_Control_88 = score_88_nSales.merge(score_88_nCust, on=['Store1','Store2'])
score_Control_88
```

	Store1	Store2	scoreNSales	scoreNcust
0	88.0	1.0	0.521294	0.622816
1	88.0	2.0	0.536997	0.623718
2	88.0	3.0	0.844619	0.868572
3	88.0	4.0	0.933564	0.946820
4	88.0	5.0	0.781275	0.833630
...	...	...	...	...
263	88.0	268.0	0.493297	0.610945
264	88.0	269.0	0.819365	0.888881
265	88.0	270.0	0.808128	0.859570
266	88.0	271.0	0.802864	0.874531
267	88.0	272.0	0.616297	0.638333

268 rows × 4 columns

```
score_Control_88['finalControlScore'] = score_Control_88.apply(lambda row: row['scoreNSales']*0.5 + row['scoreNcust']*0.5, axis=1)
score_Control_88
```

	Store1	Store2	scoreNSales	scoreNcust	finalControlScore
0	88.0	1.0	0.521294	0.622816	0.572055
1	88.0	2.0	0.536997	0.623718	0.580357
2	88.0	3.0	0.844619	0.868572	0.856596
3	88.0	4.0	0.933564	0.946820	0.940192
4	88.0	5.0	0.781275	0.833630	0.807452
...	...	...	...	...	...
263	88.0	268.0	0.493297	0.610945	0.552121
264	88.0	269.0	0.819365	0.888881	0.854123
265	88.0	270.0	0.808128	0.859570	0.833849
266	88.0	271.0	0.802864	0.874531	0.838698
267	88.0	272.0	0.616297	0.638333	0.627315

268 rows × 5 columns

Get control store

```
score_Control_88['finalControlScore'].max()
```

```
0.9604642253327091
```

```
score_Control_88.loc[score_Control_88['finalControlScore']==0.9604642253327091]
```

	Store1	Store2	scoreNSales	scoreNcust	finalControlScore
232	88.0	237.0	0.953594	0.967334	0.960464

Visualization control store

```
Time88 = measureOverTime
Time88['Store_type'] = measureOverTime88.apply(lambda row: 'Trail' if row['STORE_NBR']==88 else ('Control' if row['STORE_NBR']==2
Time88
```

	STORE_NBR	year_month	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	Store_type
0	1	2018-06-01	4.8	2	1.000000	1.000000	2.400000	Other stores
1	1	2018-07-01	220.2	52	1.057692	1.181818	3.387692	Other stores
2	1	2018-08-01	171.8	41	1.000000	1.292683	3.241509	Other stores
3	1	2018-09-01	278.2	59	1.050847	1.209677	3.709333	Other stores
4	1	2018-10-01	186.4	44	1.022727	1.266667	3.270175	Other stores
...	...	...	...	...	...	...	...	...
3382	272	2019-02-01	395.5	45	1.066667	1.895833	4.346154	Other stores
3383	272	2019-03-01	478.5	52	1.115385	1.913793	4.310811	Other stores
3384	272	2019-04-01	415.5	50	1.040000	1.865385	4.283505	Other stores
3385	272	2019-05-01	322.8	36	1.138889	1.780488	4.421918	Other stores
3386	272	2019-06-01	297.3	32	1.093750	1.885714	4.504545	Other stores

3387 rows × 8 columns

```
measureOverTimeSales88 = measureOverTime88.groupby(['year_month','Store_type'])['totSales'].mean().reset_index()
measureOverTimeSales88
```

	year_month	Store_type	totSales
0	2018-06-01	Control	36.800000
1	2018-06-01	Other stores	24.372350
2	2018-06-01	Trail	46.600000
3	2018-07-01	Control	1460.800000
4	2018-07-01	Other stores	615.458333
5	2018-07-01	Trail	1308.200000
6	2018-08-01	Control	1388.400000
7	2018-08-01	Other stores	593.066985
8	2018-08-01	Trail	1330.800000
9	2018-09-01	Control	1303.600000
10	2018-09-01	Other stores	603.936015

```
measureOverTimeSales88.set_index('year_month',inplace=True)
pastSales88 = measureOverTimeSales88.loc['2018-06-01':'2019-01-01'].reset_index()
pastSales88
```

	year_month	Store_type	totSales
0	2018-06-01	Control	36.800000
1	2018-06-01	Other stores	24.372350
2	2018-06-01	Trail	46.600000
3	2018-07-01	Control	1460.800000
4	2018-07-01	Other stores	615.458333
5	2018-07-01	Trail	1308.200000
6	2018-08-01	Control	1388.400000
7	2018-08-01	Other stores	593.066985
8	2018-08-01	Trail	1330.800000
9	2018-09-01	Control	1303.600000
10	2018-09-01	Other stores	603.936015
11	2018-09-01	Trail	1424.000000
12	2018-10-01	Control	1297.100000
13	2018-10-01	Other stores	616.098479
14	2018-10-01	Trail	1342.800000
15	2018-11-01	Control	1434.000000
16	2018-11-01	Other stores	600.782061
17	2018-11-01	Trail	1362.800000
18	2018-12-01	Control	1256.600000
19	2018-12-01	Other stores	631.856322
20	2018-12-01	Trail	1335.800000
21	2019-01-01	Control	1222.100000
22	2019-01-01	Other stores	615.294636
23	2019-01-01	Trail	1286.600000

```
px.line(data_frame=pastSales88, x='year_month', y='totSales', color='Store_type', title='Total sales by month',labels={'year_month
```

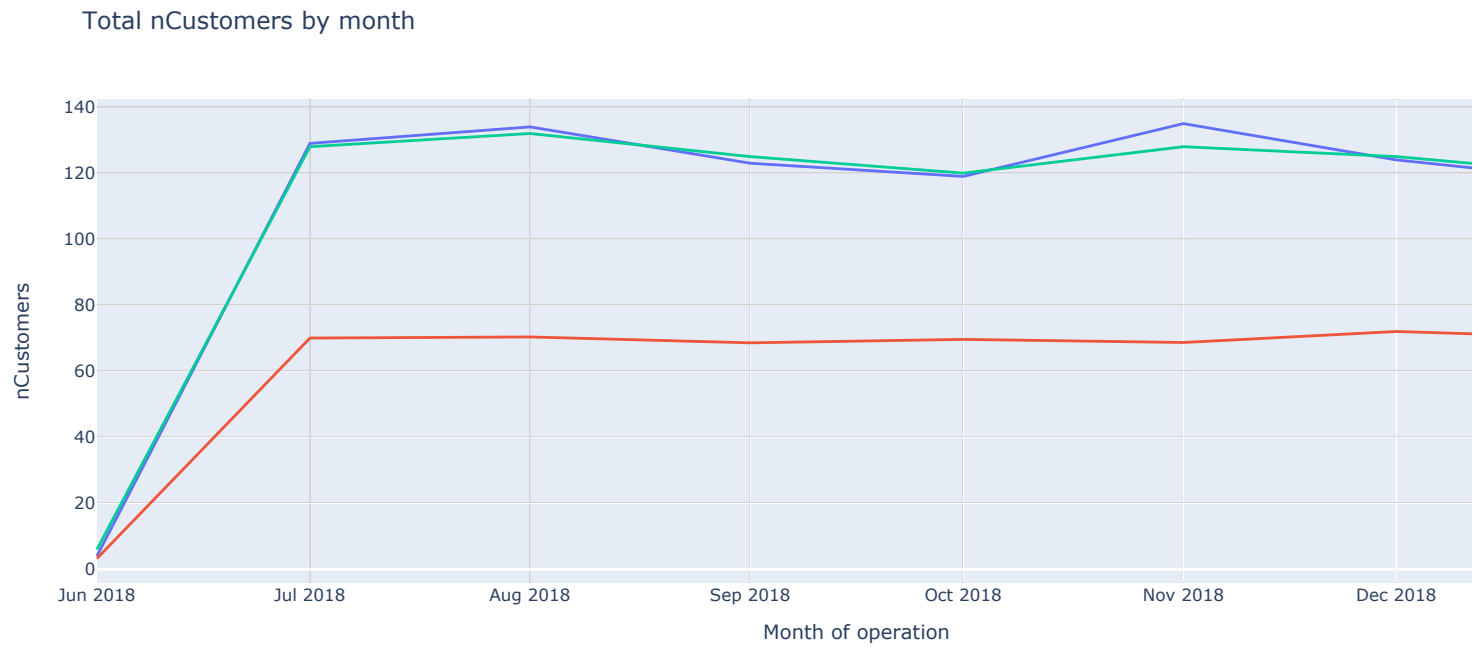

```
measureOverTimeCust88 = measureOverTime88.groupby(['year_month','Store_type'])['nCustomers'].mean().reset_index()
measureOverTimeCust88.set_index('year_month',inplace=True)
measureOverTimeCust88
```

year_month	Store_type	nCustomers
2018-06-01	Control	4.000000
2018-06-01	Other stores	3.271889
2018-06-01	Trail	6.000000
2018-07-01	Control	129.000000
2018-07-01	Other stores	70.049242
2018-07-01	Trail	128.000000
2018-08-01	Control	134.000000
2018-08-01	Other stores	70.377863
2018-08-01	Trail	132.000000
2018-09-01	Control	123.000000
2018-09-01	Other stores	68.597701
2018-09-01	Trail	125.000000
2018-10-01	Control	119.000000
2018-10-01	Other stores	69.642586
2018-10-01	Trail	120.000000
2018-11-01	Control	135.000000
2018-11-01	Other stores	68.698473
2018-11-01	Trail	128.000000
2018-12-01	Control	124.000000
2018-12-01	Other stores	72.034483
2018-12-01	Trail	125.000000
2019-01-01	Control	117.000000
2019-01-01	Other stores	69.984674
2019-01-01	Trail	119.000000
2019-02-01	Control	125.000000
2019-02-01	Other stores	64.748092
2019-02-01	Trail	124.000000
2019-03-01	Control	118.000000
2019-03-01	Other stores	70.760456
2019-03-01	Trail	134.000000
2019-04-01	Control	117.000000
2019-04-01	Other stores	68.553435
2019-04-01	Trail	128.000000
2019-05-01	Control	132.000000
2019-05-01	Other stores	70.318008
2019-05-01	Trail	131.000000
2019-06-01	Control	113.000000
2019-06-01	Other stores	66.820611

```
pastCust88 = measureOverTimeCust88.loc['2018-06-01':'2019-01-01'].reset_index()
pastCust88
```

	year_month	Store_type	nCustomers
0	2018-06-01	Control	4.000000
1	2018-06-01	Other stores	3.271889
2	2018-06-01	Trail	6.000000
3	2018-07-01	Control	129.000000
4	2018-07-01	Other stores	70.049242
5	2018-07-01	Trail	128.000000
6	2018-08-01	Control	134.000000
7	2018-08-01	Other stores	70.377863
8	2018-08-01	Trail	132.000000
9	2018-09-01	Control	123.000000
10	2018-09-01	Other stores	68.597701
11	2018-09-01	Trail	125.000000
12	2018-10-01	Control	119.000000
13	2018-10-01	Other stores	69.642586
14	2018-10-01	Trail	120.000000
15	2018-11-01	Control	135.000000
16	2018-11-01	Other stores	68.698473
17	2018-11-01	Trail	128.000000
18	2018-12-01	Control	124.000000
19	2018-12-01	Other stores	72.034483
20	2018-12-01	Trail	125.000000
21	2019-01-01	Control	117.000000
22	2019-01-01	Other stores	69.984674
23	2019-01-01	Trail	119.000000

```
px.line(data_frame=pastCust88, x='year_month', y='nCustomers', color='Store_type', title='Total nCustomers by month',labels={'yea
```



Assessment of trial period

Calculate for totSales

Scale sales

```
scalingFactorForControlSales88 = preTrialMeasures.loc[preTrialMeasures['STORE_NBR']==88, 'TOT_SALES'].sum() / preTrialMeasures.loc
scalingFactorForControlSales88
```

```
1.0040640891971833
```

Apply scaling factor

```
scaledControlSales88 = measureOverTimeSales88.loc[measureOverTimeSales88['Store_type']=='Control','totSales'].reset_index()
scaledControlSales88
```

	year_month	totSales
0	2018-06-01	36.8
1	2018-07-01	1460.8
2	2018-08-01	1388.4
3	2018-09-01	1303.6
4	2018-10-01	1297.1
5	2018-11-01	1434.0
6	2018-12-01	1256.6
7	2019-01-01	1222.1
8	2019-02-01	1399.8
9	2019-03-01	1213.2
10	2019-04-01	1181.8
11	2019-05-01	1268.3
12	2019-06-01	1077.0

```
scaledControlSales88['scaledControlSales'] = scaledControlSales88.apply(lambda row: row['totSales']*scalingFactorForControlSales88
scaledControlSales88
```

	year_month	totSales	scaledControlSales
0	2018-06-01	36.8	36.949558
1	2018-07-01	1460.8	1466.736821
2	2018-08-01	1388.4	1394.042581
3	2018-09-01	1303.6	1308.897947
4	2018-10-01	1297.1	1302.371530
5	2018-11-01	1434.0	1439.827904
6	2018-12-01	1256.6	1261.706934
7	2019-01-01	1222.1	1227.066723
8	2019-02-01	1399.8	1405.488912
9	2019-03-01	1213.2	1218.130553
10	2019-04-01	1181.8	1186.602941
11	2019-05-01	1268.3	1273.454484
12	2019-06-01	1077.0	1081.377024

```
TrailStoreSales88 = measureOverTimeSales88.loc[measureOverTimeSales88['Store_type']=='Trail',['totSales']]
TrailStoreSales88
```

year_month	totSales
2018-06-01	46.60

```
TrailStoreSales88.columns = ['trailSales']
TrailStoreSales88
```

year_month	trailSales
2018-06-01	46.60
2018-07-01	1308.20
2018-08-01	1330.80
2018-09-01	1424.00
2018-10-01	1342.80
2018-11-01	1362.80
2018-12-01	1335.80
2019-01-01	1286.60
2019-02-01	1364.00
2019-03-01	1484.20
2019-04-01	1422.40
2019-05-01	1320.45
2019-06-01	1304.60

%Diff between scaled control & trail store

```
percentageDiff88 = scaledControlSales88.merge(TrailStoreSales88, on='year_month')
percentageDiff88
```

	year_month	totSales	scaledControlSales	trailSales
0	2018-06-01	36.8	36.949558	46.60
1	2018-07-01	1460.8	1466.736821	1308.20
2	2018-08-01	1388.4	1394.042581	1330.80
3	2018-09-01	1303.6	1308.897947	1424.00
4	2018-10-01	1297.1	1302.371530	1342.80
5	2018-11-01	1434.0	1439.827904	1362.80
6	2018-12-01	1256.6	1261.706934	1335.80
7	2019-01-01	1222.1	1227.066723	1286.60
8	2019-02-01	1399.8	1405.488912	1364.00
9	2019-03-01	1213.2	1218.130553	1484.20
10	2019-04-01	1181.8	1186.602941	1422.40
11	2019-05-01	1268.3	1273.454484	1320.45
12	2019-06-01	1077.0	1081.377024	1304.60

```
percentageDiff88['percentDiff'] = percentageDiff88.apply(lambda row: (row['scaledControlSales']-row['trailSales'])/row['scaledCont
percentageDiff88
```

	year_month	totSales	scaledControlSales	trailSales	percentDiff
0	2018-06-01	36.8	36.949558	46.60	-0.261179
1	2018-07-01	1460.8	1466.736821	1308.20	0.108088
2	2018-08-01	1388.4	1394.042581	1330.80	0.045366

Get standard deviation

4	2018-10-01	1297.1	1302.371530	1342.80	-0.031042

```
stdDev = percentageDiff88.loc[percentageDiff88['year_month']< '2019-02-01', 'percentDiff'].std(ddof=8-1)
stdDev
```

```
0.29914100305358127
```

Calculate t-values

```
percentageDiff88['t-value'] = percentageDiff88.apply(lambda row: (row['percentDiff']- 0) / stdDev,axis=1)
percentageDiff88
```

	year_month	totSales	scaledControlSales	trailSales	percentDiff	t-value
0	2018-06-01	36.8	36.949558	46.60	-0.261179	-0.873096
1	2018-07-01	1460.8	1466.736821	1308.20	0.108088	0.361328
2	2018-08-01	1388.4	1394.042581	1330.80	0.045366	0.151655
3	2018-09-01	1303.6	1308.897947	1424.00	-0.087938	-0.293969
4	2018-10-01	1297.1	1302.371530	1342.80	-0.031042	-0.103771
5	2018-11-01	1434.0	1439.827904	1362.80	0.053498	0.178839
6	2018-12-01	1256.6	1261.706934	1335.80	-0.058724	-0.196310
7	2019-01-01	1222.1	1227.066723	1286.60	-0.048517	-0.162187
8	2019-02-01	1399.8	1405.488912	1364.00	0.029519	0.098680
9	2019-03-01	1213.2	1218.130553	1484.20	-0.218424	-0.730172
10	2019-04-01	1181.8	1186.602941	1422.40	-0.198716	-0.664289
11	2019-05-01	1268.3	1273.454484	1320.45	-0.036904	-0.123366
12	2019-06-01	1077.0	1081.377024	1304.60	-0.206425	-0.690058

95th & 5th percentile of control store

```
pastSales_88_Controls95 = measureOverTimeSales88.loc[measureOverTimeSales88['Store_type']=='Control']
pastSales_88_Controls95['totSales'] = pastSales_88_Controls95.apply(lambda row: row['totSales']*(1+stdDev*2),axis=1)
pastSales_88_Controls95.iloc[0:13,0] = 'Control 95th % confidence interval'
pastSales_88_Controls95.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:
```

```
pastSales_88_Controls5 = measureOverTimeSales88.loc[measureOverTimeSales88['Store_type']=='Control']
pastSales_88_Controls5['totSales'] = pastSales_88_Controls5.apply(lambda row: row['totSales']*(1-stdDev*2),axis=1)
pastSales_88_Controls5.iloc[0:13,0] = 'Control 5th % confidence interval'
pastSales_88_Controls5.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
```

	year_month	Store_type	totSales
0	2018-06-01	Control 5th % confidence interval	14.783222
1	2018-07-01	Control 5th % confidence interval	586.829645
2	2018-08-01	Control 5th % confidence interval	557.745263
3	2018-09-01	Control 5th % confidence interval	523.679577
4	2018-10-01	Control 5th % confidence interval	521.068410
5	2018-11-01	Control 5th % confidence interval	576.063603
6	2018-12-01	Control 5th % confidence interval	504.798831
7	2019-01-01	Control 5th % confidence interval	490.939560
8	2019-02-01	Control 5th % confidence interval	562.324848
9	2019-03-01	Control 5th % confidence interval	487.364270
10	2019-04-01	Control 5th % confidence interval	474.750325
11	2019-05-01	Control 5th % confidence interval	509.498932
12	2019-06-01	Control 5th % confidence interval	432.650279

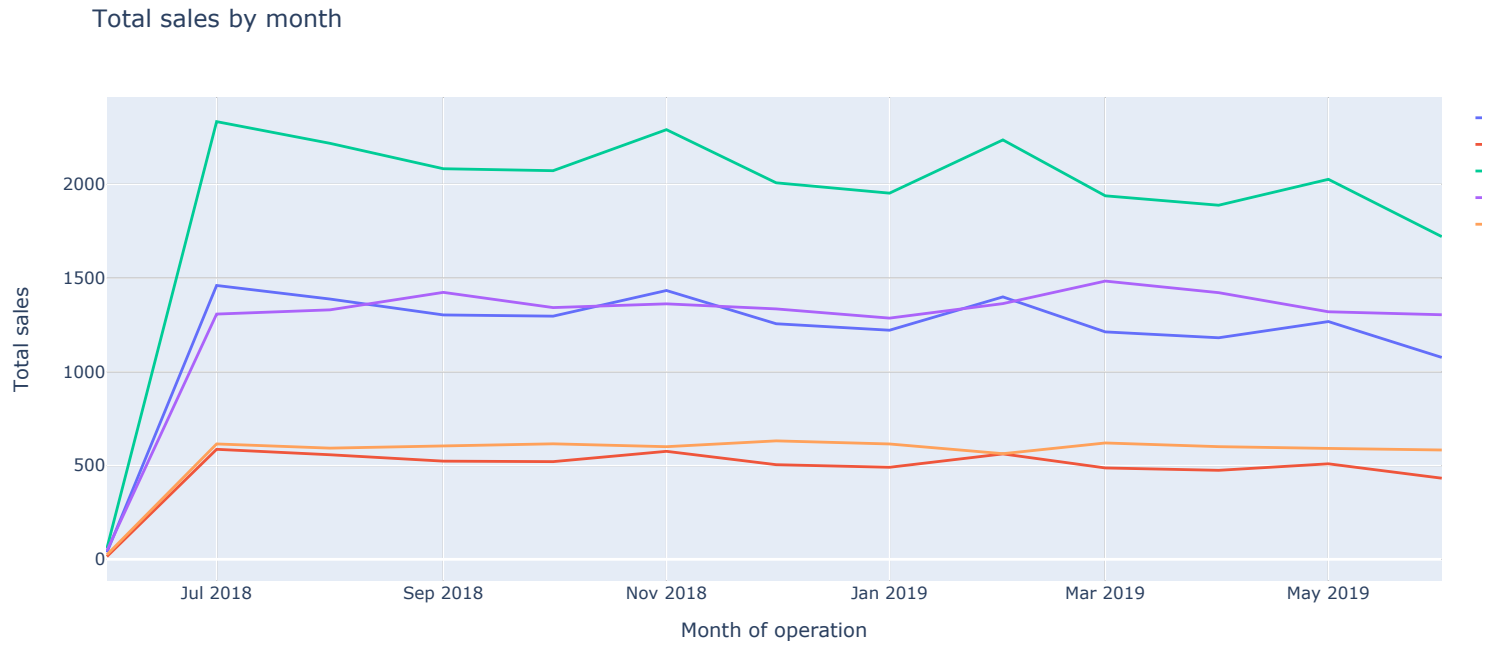
```
trialAssessment_sales_88 = pd.concat([measureOverTimeSales88,pastSales_88_Controls5,pastSales_88_Controls95])
trialAssessment_sales_88 = trialAssessment_sales_88.sort_values(by=['year_month'])
trialAssessment_sales_88 = trialAssessment_sales_88.reset_index()
trialAssessment_sales_88
```

	year_month	Store_type	totSales
0	2018-06-01	Control	36.800000
1	2018-06-01	Control 5th % confidence interval	14.783222
2	2018-06-01	Control 95th % confidence interval	58.816778
3	2018-06-01	Trail	46.600000
4	2018-06-01	Other stores	24.372350
...	...	...	...
60	2019-06-01	Trail	1304.600000
61	2019-06-01	Other stores	583.157634
62	2019-06-01	Control	1077.000000
63	2019-06-01	Control 5th % confidence interval	432.650279
64	2019-06-01	Control 95th % confidence interval	1721.349721

65 rows × 3 columns

Visualization Trial

```
nt_sales_88, x='year_month', y='totSales', color='Store_type', title='Total sales by month',labels={'year_month':'Month of operat
```



Calculate for nCustomers

Scale nCustomers

```
scalingFactorForControlNcust88 = preTrialMeasures.loc[preTrialMeasures['STORE_NBR']==88, 'nCustomers'].sum() / preTrialMeasures.lo
scalingFactorForControlNcust88
```

```
0.9977401129943503
```

Apply scaling factor

```
scaledControlNcust88 = measureOverTimeCust88.loc[measureOverTimeCust88['Store_type']=='Control','nCustomers'].reset_index()
scaledControlNcust88
```

	year_month	nCustomers
0	2018-06-01	4.0
1	2018-07-01	129.0
2	2018-08-01	134.0
3	2018-09-01	123.0
4	2018-10-01	119.0
5	2018-11-01	135.0
6	2018-12-01	124.0
7	2019-01-01	117.0
8	2019-02-01	125.0
9	2019-03-01	118.0
10	2019-04-01	117.0
11	2019-05-01	132.0
12	2019-06-01	113.0

```
scaledControlNcust88['scaledControlNcust'] = scaledControlNcust88.apply(lambda row: row['nCustomers']*scalingFactorForControlNcust
scaledControlNcust88
```

	year_month	nCustomers	scaledControlNcust
0	2018-06-01	4.0	3.990960
1	2018-07-01	129.0	128.708475
2	2018-08-01	134.0	133.697175
3	2018-09-01	123.0	122.722034
4	2018-10-01	119.0	118.731073
5	2018-11-01	135.0	134.694915
6	2018-12-01	124.0	123.719774
7	2019-01-01	117.0	116.735593

```
TrailStoreNcust88 = measureOverTimeCust88.loc[measureOverTimeCust88['Store_type']=='Trail',['nCustomers']]
TrailStoreNcust88
```

year_month	nCustomers
2018-06-01	6.0
2018-07-01	128.0
2018-08-01	132.0
2018-09-01	125.0
2018-10-01	120.0
2018-11-01	128.0
2018-12-01	125.0
2019-01-01	119.0
2019-02-01	124.0
2019-03-01	134.0
2019-04-01	128.0
2019-05-01	131.0
2019-06-01	119.0

```
TrailStoreNcust88.columns = ['trailNcust']
TrailStoreNcust88
```

year_month	trailNcust
2018-06-01	6.0
2018-07-01	128.0
2018-08-01	132.0
2018-09-01	125.0
2018-10-01	120.0
2018-11-01	128.0
2018-12-01	125.0
2019-01-01	119.0
2019-02-01	124.0
2019-03-01	134.0
2019-04-01	128.0
2019-05-01	131.0
2019-06-01	119.0

%Diff between scaled control & trail store

```
percentageDiff88 = scaledControlNcust88.merge(TrailStoreNcust88, on='year_month')
percentageDiff88
```

	year_month	nCustomers	scaledControlNcust	trailNcust
0	2018-06-01	4.0	3.990960	6.0
1	2018-07-01	129.0	128.708475	128.0
2	2018-08-01	134.0	133.697175	132.0
3	2018-09-01	123.0	122.722034	125.0
4	2018-10-01	119.0	118.731073	120.0
5	2018-11-01	135.0	134.694915	128.0
6	2018-12-01	124.0	123.719774	125.0
7	2019-01-01	117.0	116.735593	119.0
8	2019-02-01	125.0	124.717514	124.0

```
percentageDiff88['percentDiff'] = percentageDiff88.apply(lambda row: (row['scaledControlNcust']-row['trailNcust'])/row['scaledCont
percentageDiff88
```

	year_month	nCustomers	scaledControlNcust	trailNcust	percentDiff
0	2018-06-01	4.0	3.990960	6.0	-0.503398
1	2018-07-01	129.0	128.708475	128.0	0.005504
2	2018-08-01	134.0	133.697175	132.0	0.012694
3	2018-09-01	123.0	122.722034	125.0	-0.018562
4	2018-10-01	119.0	118.731073	120.0	-0.010687
5	2018-11-01	135.0	134.694915	128.0	0.049704
6	2018-12-01	124.0	123.719774	125.0	-0.010348
7	2019-01-01	117.0	116.735593	119.0	-0.019398
8	2019-02-01	125.0	124.717514	124.0	0.005753
9	2019-03-01	118.0	117.733333	134.0	-0.138165
10	2019-04-01	117.0	116.735593	128.0	-0.096495
11	2019-05-01	132.0	131.701695	131.0	0.005328
12	2019-06-01	113.0	112.744633	119.0	-0.055483

Get standard deviation

```
stdDev = percentageDiff88.loc[percentageDiff88['year_month']< '2019-02-01', 'percentDiff'].std(ddof=8-1)
stdDev
```

```
0.4758656807356589
```

Calculate t-values

```
percentageDiff88['t-value'] = percentageDiff88.apply(lambda row: (row['percentDiff']- 0) / stdDev,axis=1)
percentageDiff88
```

	year_month	nCustomers	scaledControlNcust	trailNcust	percentDiff	t-value
0	2018-06-01	4.0	3.990960	6.0	-0.503398	-1.057856
1	2018-07-01	129.0	128.708475	128.0	0.005504	0.011567
2	2018-08-01	134.0	133.697175	132.0	0.012694	0.026676
3	2018-09-01	123.0	122.722034	125.0	-0.018562	-0.039007
4	2018-10-01	119.0	118.731073	120.0	-0.010687	-0.022459
5	2018-11-01	135.0	134.694915	128.0	0.049704	0.104450
6	2018-12-01	124.0	123.719774	125.0	-0.010348	-0.021745
7	2019-01-01	117.0	116.735593	119.0	-0.019398	-0.040763
8	2019-02-01	125.0	124.717514	124.0	0.005753	0.012090
9	2019-03-01	118.0	117.733333	134.0	-0.138165	-0.290345
10	2019-04-01	117.0	116.735593	128.0	-0.096495	-0.202778
11	2019-05-01	132.0	131.701695	131.0	0.005328	0.011196
12	2019-06-01	113.0	112.744633	119.0	-0.055483	-0.116593

95th & 5th percentile of control store

```
pastCust_88_Controls95 = measureOverTimeCust88.loc[measureOverTimeCust88['Store_type']=='Control']
pastCust_88_Controls95['nCustomers'] = pastCust_88_Controls95.apply(lambda row: row['nCustomers']*(1+stdDev*2),axis=1)
pastCust_88_Controls95.iloc[0:13,0] = 'Control 95th % confidence interval'
pastCust_88_Controls95.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
```

	year_month	Store_type	nCustomers
0	2018-06-01	Control 95th % confidence interval	7.806925
1	2018-07-01	Control 95th % confidence interval	251.773346
2	2018-08-01	Control 95th % confidence interval	261.532002
3	2018-09-01	Control 95th % confidence interval	240.062957
4	2018-10-01	Control 95th % confidence interval	232.256032
5	2018-11-01	Control 95th % confidence interval	263.483734
6	2018-12-01	Control 95th % confidence interval	242.014689
7	2019-01-01	Control 95th % confidence interval	228.352569
8	2019-02-01	Control 95th % confidence interval	243.966420
9	2019-03-01	Control 95th % confidence interval	230.304301
10	2019-04-01	Control 95th % confidence interval	228.352569
11	2019-05-01	Control 95th % confidence interval	257.628540
12	2019-06-01	Control 95th % confidence interval	220.545644

```
pastCust_88_Controls5 = measureOverTimeCust88.loc[measureOverTimeCust88['Store_type']=='Control']
pastCust_88_Controls5['nCustomers'] = pastCust_88_Controls5.apply(lambda row: row['nCustomers']*(1-stdDev*2),axis=1)
pastCust_88_Controls5.iloc[0:13,0] = 'Control 5th % confidence interval'
pastCust_88_Controls5.reset_index()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

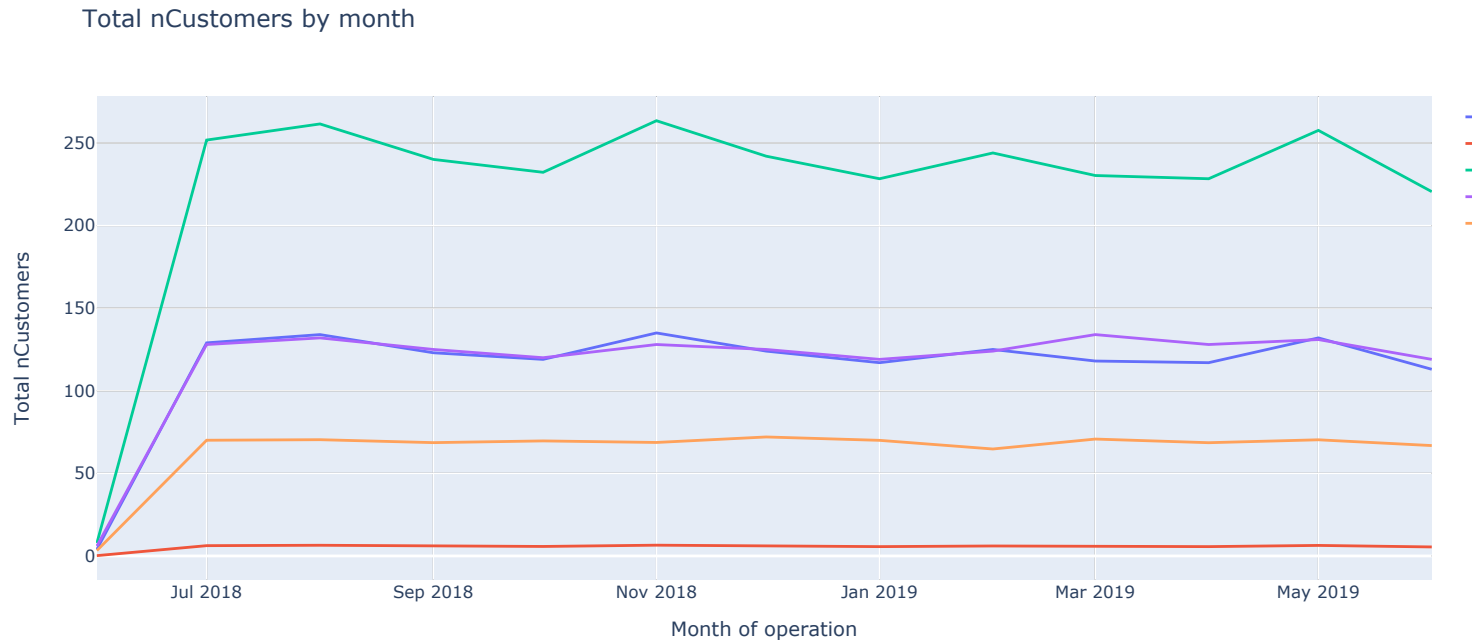
```
trialAssessment_cust_88 = pd.concat([measureOverTimeCust88,pastCust_88_Controls5,pastCust_88_Controls95])
trialAssessment_cust_88 = trialAssessment_cust_88.sort_values(by=['year_month'])
trialAssessment_cust_88 = trialAssessment_cust_88.reset_index()
trialAssessment_cust_88
```

	year_month	Store_type	nCustomers
0	2018-06-01	Control	4.000000
1	2018-06-01	Control 5th % confidence interval	0.193075
2	2018-06-01	Control 95th % confidence interval	7.806925
3	2018-06-01	Trail	6.000000
4	2018-06-01	Other stores	3.271889
...	...	...	...
60	2019-06-01	Trail	119.000000
61	2019-06-01	Other stores	66.820611
62	2019-06-01	Control	113.000000
63	2019-06-01	Control 5th % confidence interval	5.454356
64	2019-06-01	Control 95th % confidence interval	220.545644

65 rows × 3 columns

Visualization Trial

```
px.line(data_frame=trialAssessment_cust_88, x='year_month', y='nCustomers', color='Store_type', title='Total nCustomers by month',
```



Conclusion

The results for trial stores 77 and 88 during the trial period show a significant difference in at least two of the three period.