In [343]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import seaborn as sns
from scipy import stats
from datetime import datetime
sns.set_theme('notebook')
from functools import reduce
```

Experimentation and uplift testing

by Murong (Sophie) Cui

Identify benchmark stores that allow us to test the impact of the trial store layouts on cutomer sales and evaluate the performance of a store trial which was performed in stores 77, 86 and 88. The trial period was Fed 2019 to April 2019.

Table of content

Introduction

During the experimentation, we test the impact of the new trial layouts with a data driven recommendation to whether or not the trial layout should be rolled out to all their stores. By examining the performance in trial vs control stores, the experimentation provides a recommendation for each location based on the insight.

Firstly, we select control stores. Exploring the data and defining metrics for the control store selection, we take total sales revenue and total number of customer into account. When we consider the monthly sales experience of each store.

Then, we conduct assessment of the trial. This assessment should give us insights into each of the store by check each trial individually in comparsion with the control store to get a clear view of its overall performance, which helps us to determine wheather the trial stores were successful or not.

3+.At the end, we summarize our findings for each store and provide an recommendation that we can share with Julia outlining the impact on sales during the trial period.

Dataset

`QVI_data.csv` contains a year's worth of product transactions and customer demographics.

- LYLTY_CARD_NBR: loyalty card id number
- DATE: the days after the base date (1899-12-30)
- STORE_NBR: the store ID number
- TXN_ID: transaction ID
- PROD_NBR: Product ID Number
- PROD_NAME: product name
- PROD_QTY: product quantity sold
- TOT_SALES: total sales
- PACK_SIZE: package size
- BRAND: the brand of chip
- LIFESTAGE: customer attribute that identifies whether a customer has a family or not and what point in life they are at: RETIREES, OLDER SINGLES/COUPLES, YOUNG SINGLES/COUPLES, OLDER FAMILIES, YOUNG FAMILIES, MIDAGE SINGLES/COUPLES, NEW FAMILIES.
- PREMIUM_CUSTOMER: Customer segmentation used to differentiate shoppers by the price point of products they buy and the types of products they buy. It is used too identify whether customers may spend more for quality or brand or wehter they will purchase the cheapest options. Budget, Mainstream, Premium

In [344]:
```
raw_data = pd.read_csv('data/QVI_data.csv')
df = raw_data.copy()
df['YEARMONTH'] = [''.join(x.split('-')[0:2]) for x in df['DATE']]
df['YEARMONTH'] = pd.to_datetime(df['YEARMONTH'], format = '%Y%m')
df['DATE'] = pd.to_datetime(df['DATE'])
df.head(2)
```

Out[344]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_!
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	

Data Preparation

Create a measure to compare different control stores to each of the trial stores to do this write a function to reduce having to re-do the analysis for each trial store. Using Pearson correlations or a metric such as a magnitude distance as a measure.

The client has selected store numbers 77, 86, 88 has trail stores and want control stores to be established store athat operational for the entire observation period

We would want to match trail stores to control stores that are similar to the trial store prior to the trial period of Feb 2019 in terms of:

- Monthly overall sales revenue
- Monthly number of customers

First, we define the measure calculations to use during the analysis. For each store and month calculate total sales, number of customers, transactions per customer, chips per customer and average price per unit. `measureOverTime`

Then, we create the metrics of interest and filter to stores that are present throughout the pre-trial period. `preTrialMeasures`

In [345]:
```
measureOverTime = df\
.groupby(['STORE_NBR', 'YEARMONTH'])\
.agg({'TOT_SALES': 'sum',
     'LYLTY_CARD_NBR': 'nunique',
     'TXN_ID': 'nunique',
    'PROD_QTY': 'sum'
     })\
.reset_index()
measureOverTime.columns = ['STORE_NBR', 'YEARMONTH', 'totSales', 'nCusto
mers', 'nTxn', 'totChips']

measureOverTime['nTxnPerCust'] = measureOverTime['nTxn']/measureOverTime
['nCustomers']
measureOverTime['nChipPerTxn'] = measureOverTime['totChips']/measureOver
Time['nTxn']
measureOverTime['avgPricePerUnit'] = measureOverTime['totSales']/measure
OverTime['totChips']

measureOverTime.drop(['nTxn', 'totChips'], axis = 1, inplace = True)

storesWithFullObs = measureOverTime.groupby('STORE_NBR').filter(lambda x
: x['YEARMONTH'].nunique() == 12)
preTrialMeasures = storesWithFullObs.loc[storesWithFullObs['YEARMONTH']
< datetime(2019, 2, 1)]
preTrialMeasures.head(2)
```

Out[345]:

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTxnPerCust	nChipPerTxn	avgPricePerUnit
0	1	2018-07-01	206.9	49	1.061224	1.192308	3.337097
1	1	2018-08-01	176.1	42	1.023810	1.255814	3.261111

Control Store Selection

In order to rank how similar each potential control store is to the trial store, we could calculate how correlated the performance of each store is the trial store by calculating correlation. Apartb from correlation, we could also calculate a standardise metric based on the absolute difference between the trial store's performance of each store.

After we have all the correlation and magnitude of distance for total sales and number of customers between control store and trial store, we could take simple average of all the scores calculated using our function to create a composite score to rank on.

In [346]:
```
def correlation_table_by_feature(target_store, feature, df=preTrialMeasu
res):
    corr = []
    nStore = df.STORE_NBR.nunique()
    a = df[df.STORE_NBR == target_store][feature]
    for i in df.STORE_NBR.unique():
        b = df[df.STORE_NBR == i][feature]
        cor_coe = np.corrcoef(a, b)[0, 1]
        corr.append(cor_coe)
    calcCorrTable =  pd.DataFrame({'target_store': [target_store]*nStore
,
                              'compared_store': df.STORE_NBR.unique(),
                              'correlation': corr})
    return calcCorrTable


def magnitude_distance_by_feature(target_store, feature, df=preTrialMeas
ures):
    distance = []
    for i in df.STORE_NBR.unique():
        for j in df.YEARMONTH.unique():
            feature_otherstore_yearmonth = df[(df.STORE_NBR == i) & (df.
YEARMONTH == j)][feature].values
            feature_targetstore_yearmonth = df[(df.STORE_NBR == target_s
tore)&(df.YEARMONTH == j)][feature].values
            distance.append(np.round(abs(feature_otherstore_yearmonth -
feature_targetstore_yearmonth), 3))
    distances_overtime_pair_stores = pd.DataFrame({'compared_store': df[
'STORE_NBR'].values,
                                                   'YEARMONTH': df['YEARM
ONTH'].values,
                                                   'DISTANCE': np.array(d
istance).flatten()})
    # standardise the magnitude distance so that the measure ranges from
0 to 1
    max_distance_yearmonth_allstores = distances_overtime_pair_stores.gr
oupby('YEARMONTH')['DISTANCE'].max()
    min_distance_yearmonth_allstores = distances_overtime_pair_stores.gr
oupby('YEARMONTH')['DISTANCE'].min()

    magnitude_distance = []
    for i in df.STORE_NBR.unique():
        feature_otherstore_overtime = df[df.STORE_NBR == i][feature].val
ues
        feature_targetstore_overtime = df[df.STORE_NBR == target_store][
feature].values
        nom = abs(feature_otherstore_overtime - feature_targetstore_over
time)
        measure = (1 - (nom - min_distance_yearmonth_allstores)/(max_dis
tance_yearmonth_allstores - min_distance_yearmonth_allstores)).mean()
        magnitude_distance.append(measure)
    finalDistTable = pd.DataFrame({'target_store': [target_store]*df.STO
RE_NBR.nunique(),
                                  'compared_store': df.STORE_NBR.unique
(),
                                  'magnitude_distance': magnitude_distan
```

```
ce})

    return finalDistTable

def composite_score(target_store):
    corr_weight = 0.5
    correlation_totSales = correlation_table_by_feature(target_store, 't
otSales')
    correlation_nCustomers = correlation_table_by_feature(target_store,
'nCustomers')
    magDis_totSales = magnitude_distance_by_feature(target_store, 'totSa
les')
    magDis_nCustomers = magnitude_distance_by_feature(target_store, 'nCu
stomers')

    data_frames = [correlation_totSales, correlation_nCustomers, magDis_
totSales, magDis_nCustomers]
    df_merged = reduce(lambda left, right: pd.merge(left, right, on = [
'target_store', 'compared_store'], how = 'left'), data_frames)

    df_merged['finalControlScore'] = corr_weight*(corr_weight*df_merged[
'correlation_x'] + (1 - corr_weight)*df_merged['magnitude_distance_x'])
+ (1-corr_weight)*(corr_weight*df_merged['correlation_y'] + (1 - corr_we
ight)*df_merged['magnitude_distance_y'])
    control_store = df_merged.sort_values('finalControlScore', ascending
= False).iloc[1][1].astype(int)
    return control_store
```

In [347]:
```
def df_ready_to_plot_preTrial(target_store, control_store, feature):
    conditions = [(measureOverTimeSales['STORE_NBR'] == target_store),
                  (measureOverTimeSales['STORE_NBR'] == control_store),
                  (~measureOverTimeSales['STORE_NBR'].isin([target_store, con
trol_store]))]

    values = ['Trial', 'Control', 'Other Stores']

    measureOverTimeSales['store_type'] = np.select(conditions, values)
    lineplot_ready_preTrial = measureOverTimeSales.groupby(['YEARMONTH',
'store_type'])[feature].mean().reset_index()
    lineplot_ready_preTrial = lineplot_ready_preTrial[lineplot_ready_pre
Trial['YEARMONTH'] < '2019-03-01']

    return lineplot_ready_preTrial

def plot_preTrial(feature, df, target_store, control_store):
    plt.figure(figsize = (12, 10))
    ax = sns.lineplot(data = df, x = 'YEARMONTH', y = feature, hue = 'st
ore_type');
    plt.ylim(0)
    ax.set_xticks(lineplot_ready_preTrail_by_totSales['YEARMONTH'].uniqu
e())
    ax.set_title('{} between Trial Store {} and Control Store {}'.format
(feature, target_store, control_store), fontsize = 16)
    plt.legend(title = 'Store Type', loc='lower left', labels = ['Contro
l Store '+str(control_store), 'Other Store', 'Trial Store '+str(target_s
tore)])
    plt.xticks(rotation = 45)
    sns.despine();
```

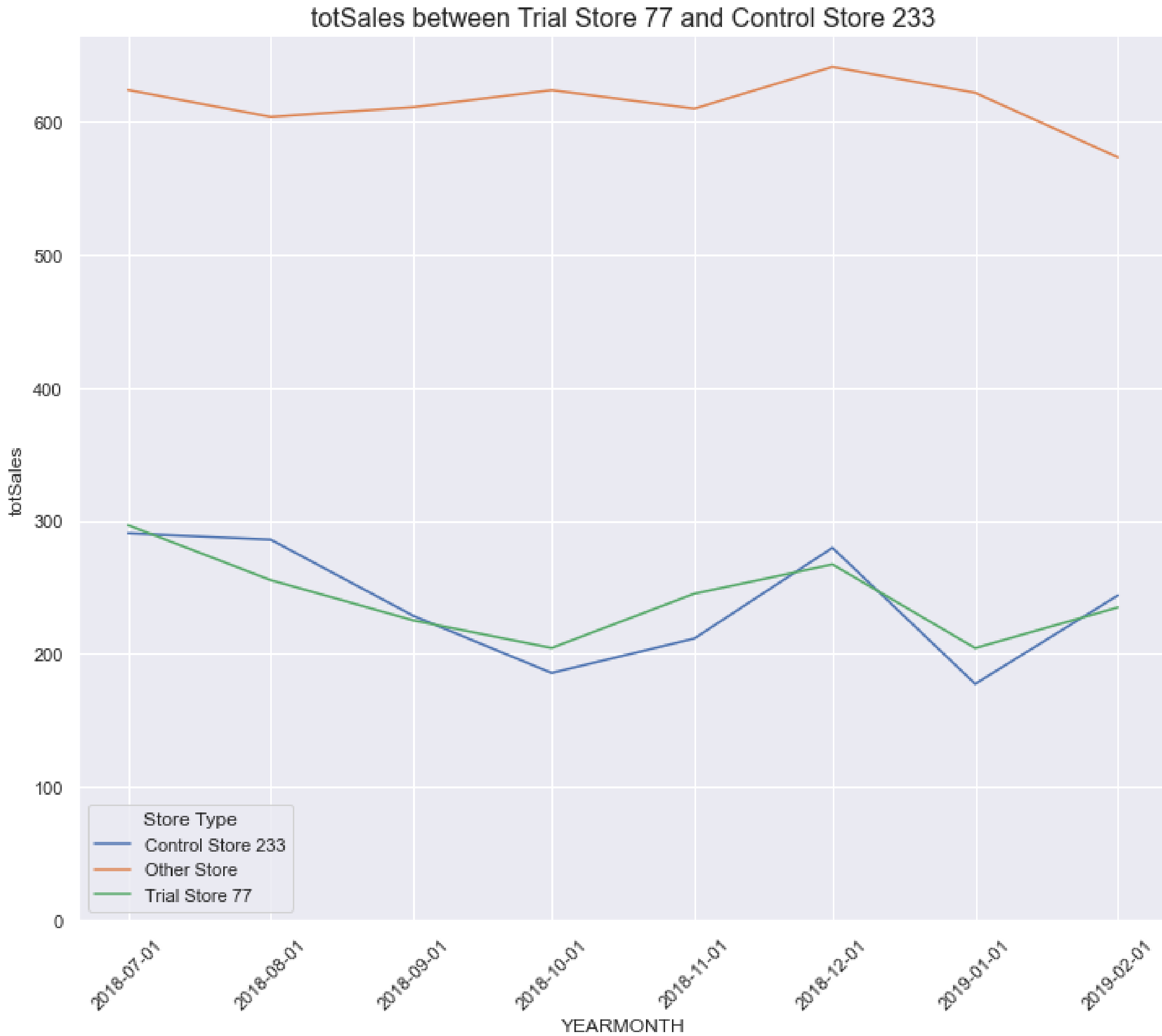
Store 77

Now use the functions to find the control stores. We'll select control stores based on how simliar monthly total sales in dollar amounts and monthly number of customers are to the trial stores. From the final similarity score, we found the control store for trial store 77 is store 233 When we found a control store, we could check visually if the drivers are indeed similar in the period before the trial period.

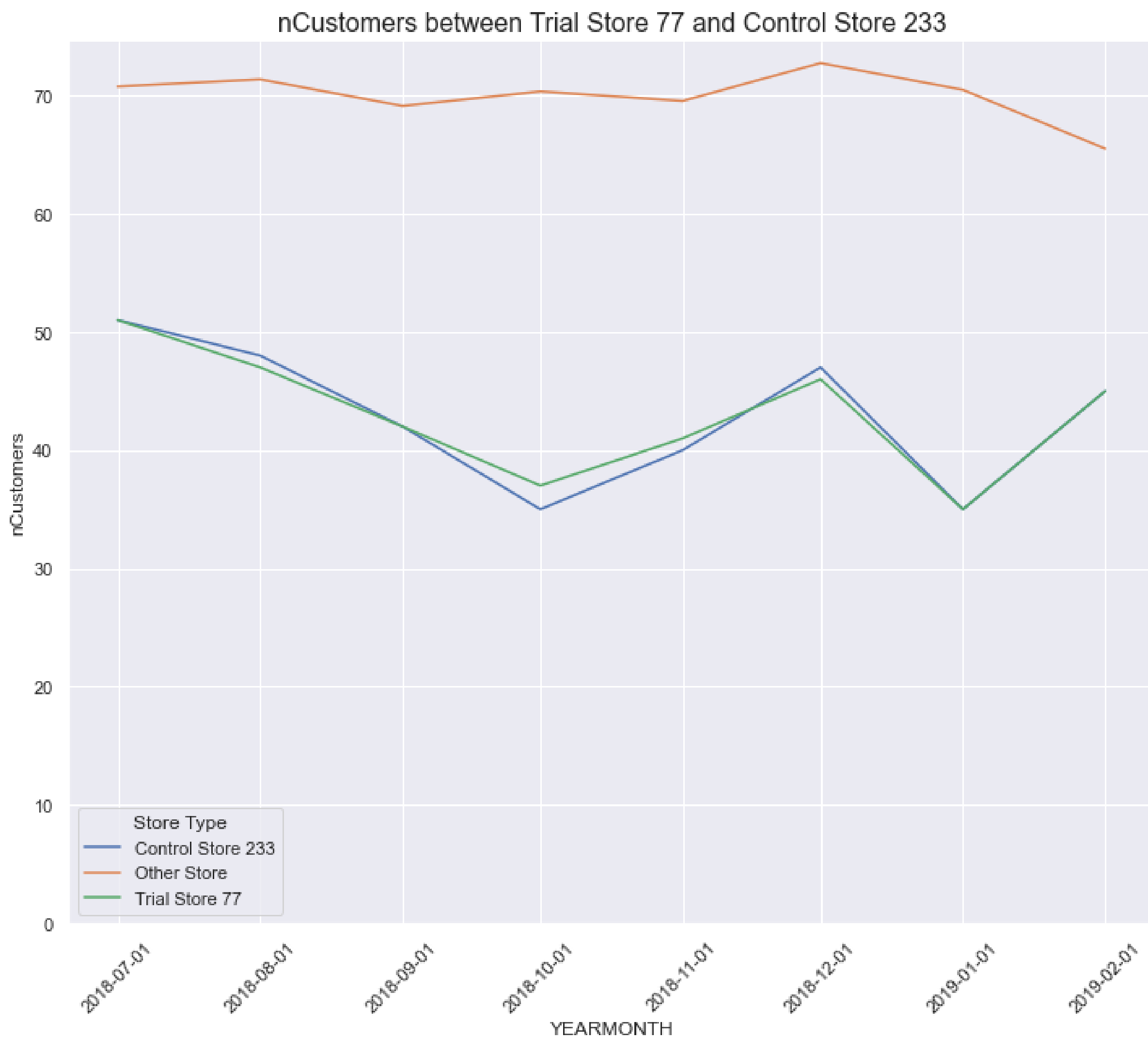
We conduct visdual checks on total sales and number of customer trends by comparing the trial store to the other store.

In [348]:
```
target_store_77 = 77
control_store_77 = composite_score(target_store_77)
```

In [349]:
```
plot_preTrial('totSales', df_ready_to_plot_preTrial(target_store_77, con
trol_store_77, 'totSales'), target_store_77, control_store_77)
```



In [350]:
```
plot_preTrial('nCustomers', df_ready_to_plot_preTrial(target_store_77, c
ontrol_store_77, 'nCustomers'), target_store_77, control_store_77)
```



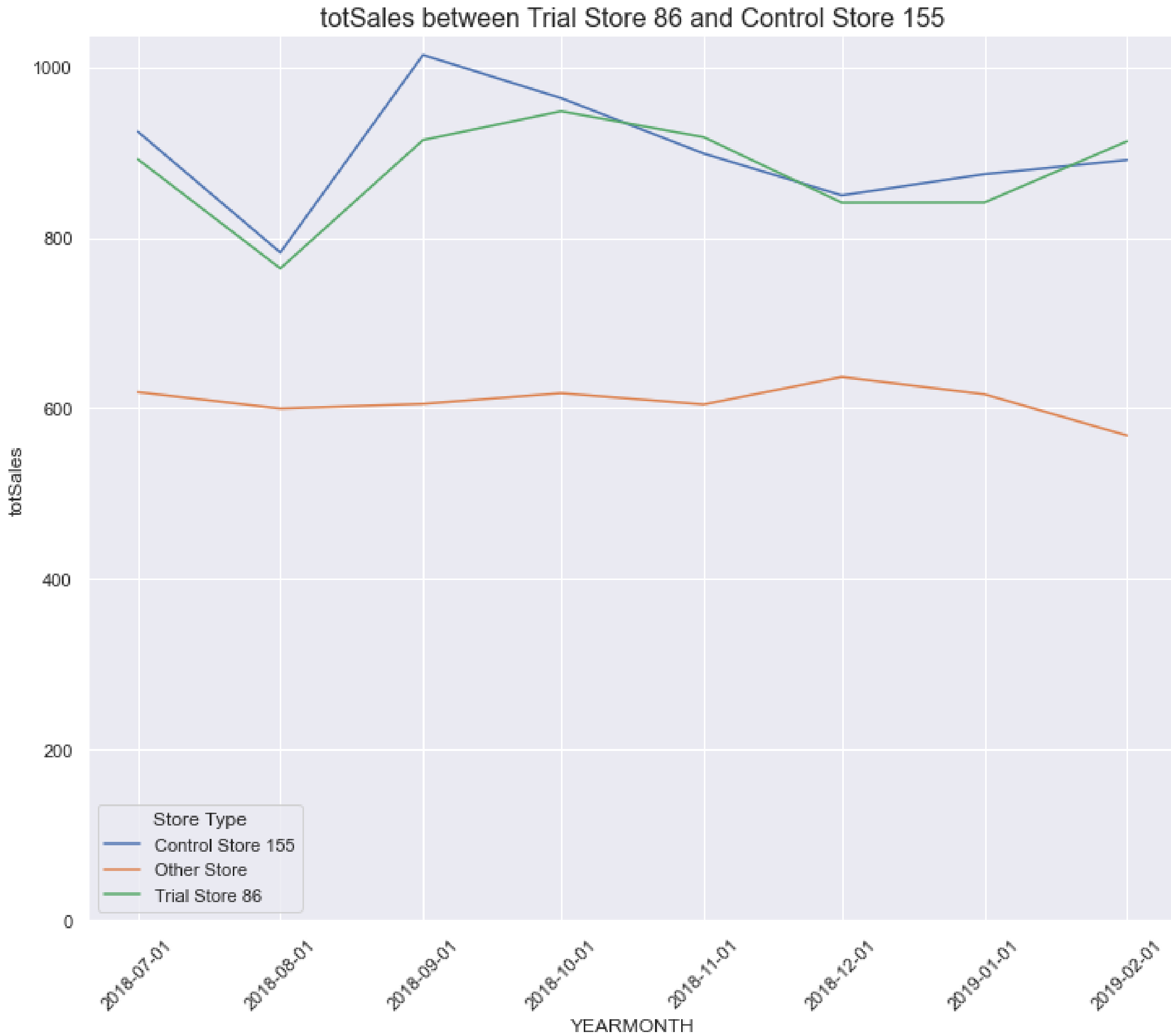
Store 86

Now use the functions to find the control stores. We'll select control stores based on how simliar monthly total sales in dollar amounts and monthly number of customers are to the trial stores. From the final similarity score, we found the control store for trial store 86 is store 155 When we found a control store, we could check visually if the drivers are indeed similar in the period before the trial period.

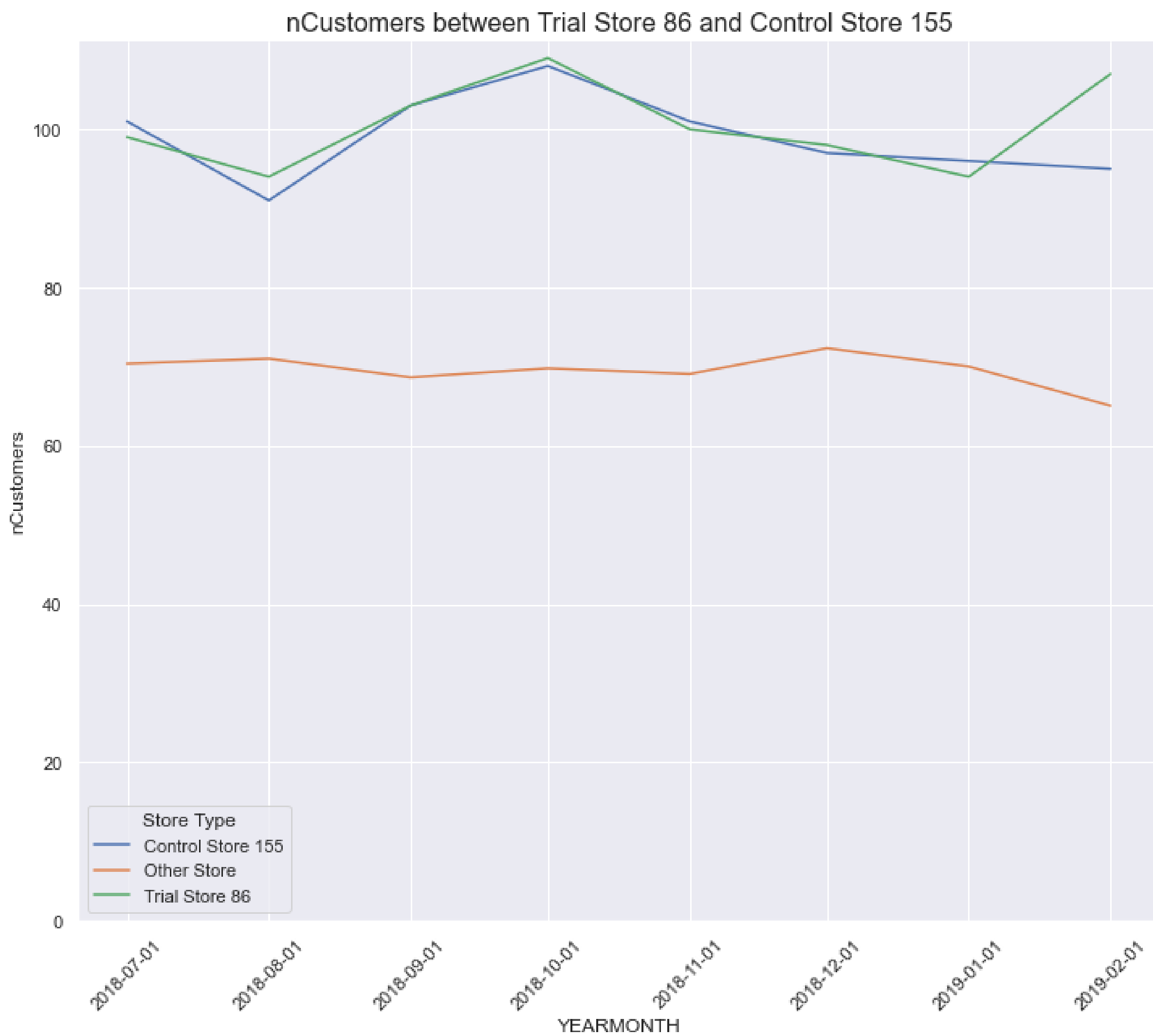
We conduct visdual checks on total sales and number of customer trends by comparing the trial store to the other store.

In [351]:
```
target_store_86 = 86
control_store_86 = composite_score(target_store_86)
```

In [352]:
```
plot_preTrial('totSales', df_ready_to_plot_preTrial(target_store_86, con
trol_store_86, 'totSales'), target_store_86, control_store_86)
```



In [353]:
```
plot_preTrial('nCustomers', df_ready_to_plot_preTrial(target_store_86, c
ontrol_store_86, 'nCustomers'), target_store_86, control_store_86)
```



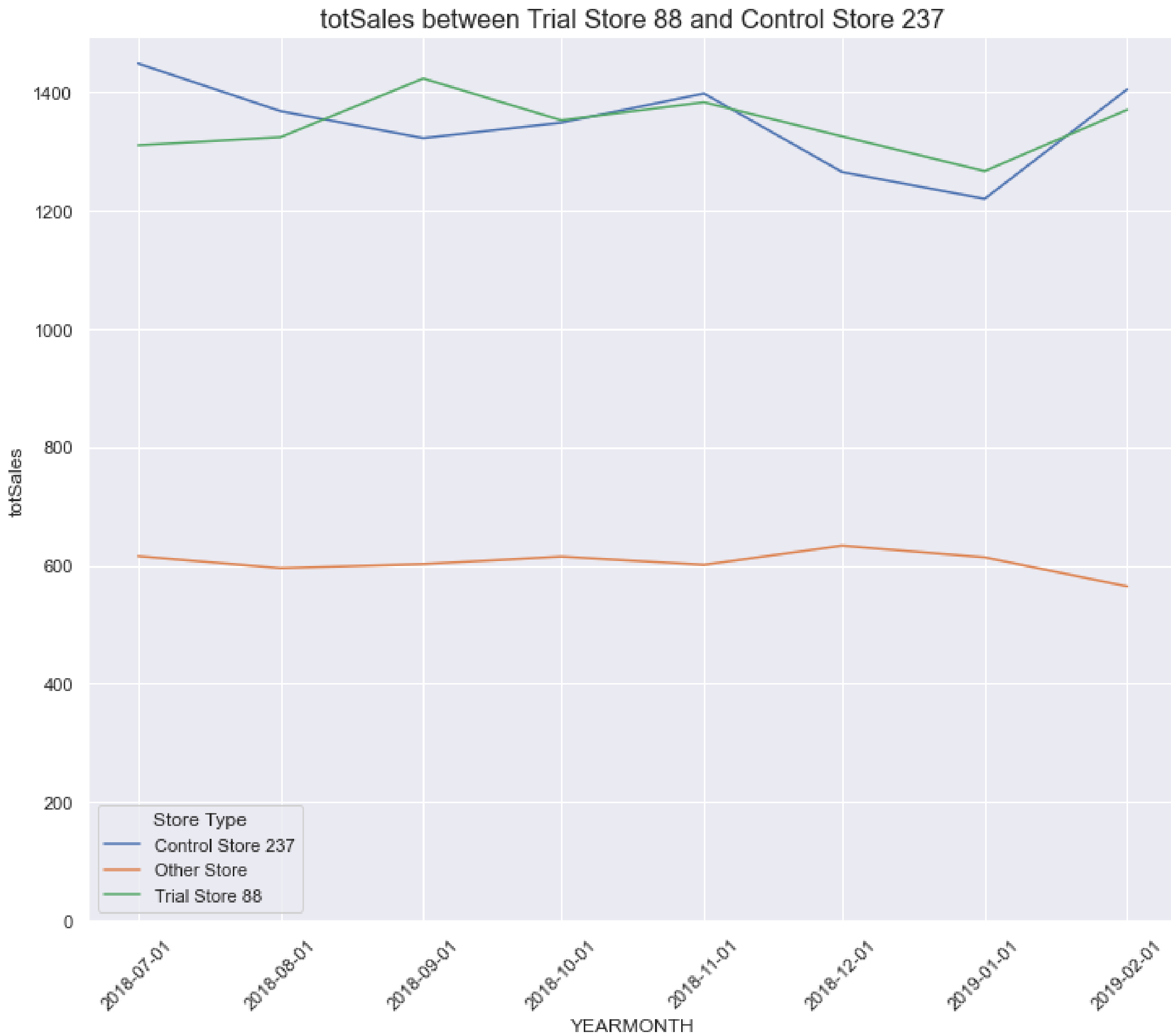
Store 88

Now use the functions to find the control stores. We'll select control stores based on how simliar monthly total sales in dollar amounts and monthly number of customers are to the trial stores. From the final similarity score, we found the control store for trial store 88 is store 237 When we found a control store, we could check visually if the drivers are indeed similar in the period before the trial period.

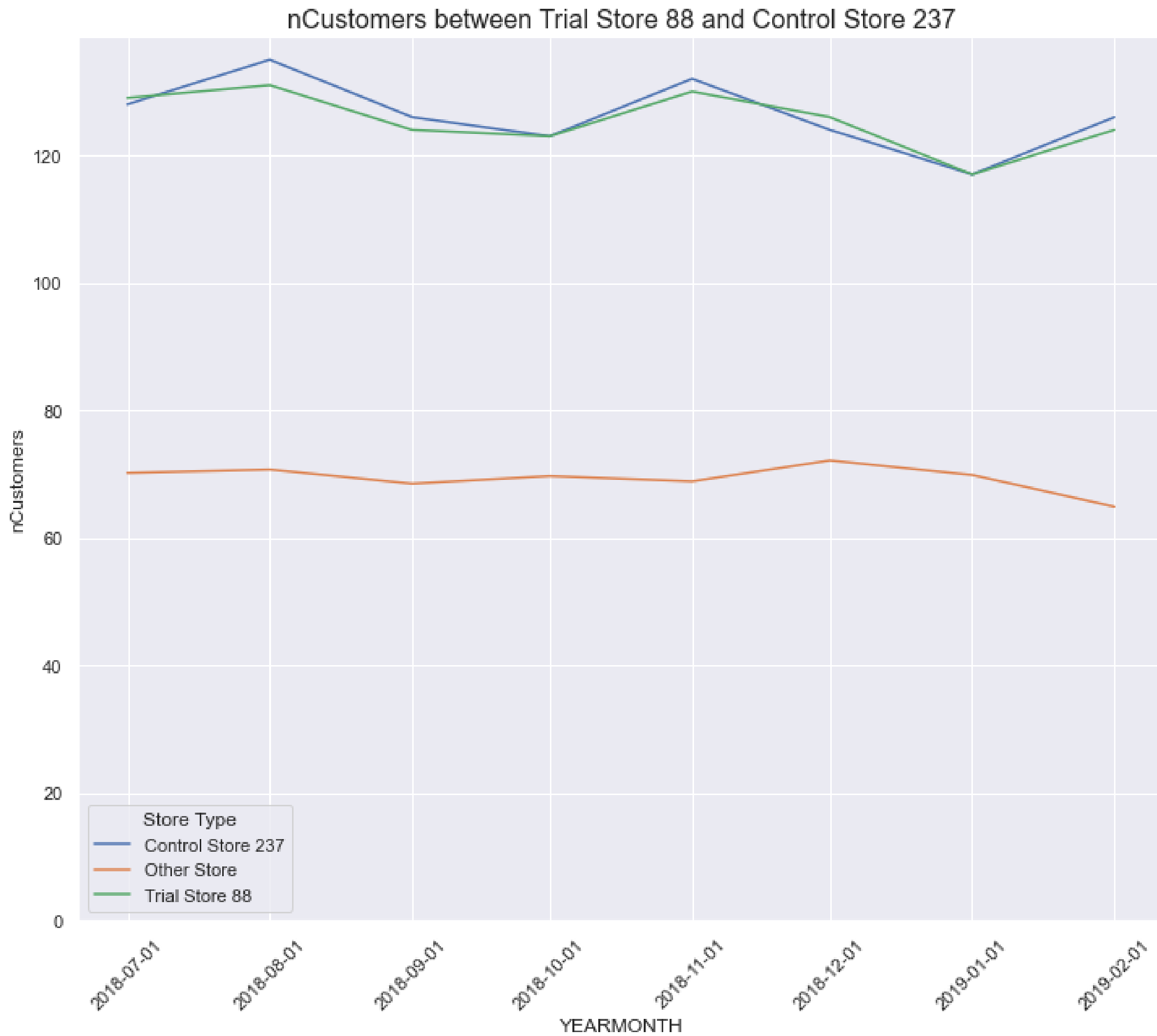
We conduct visdual checks on total sales and number of customer trends by comparing the trial store to the other store.

In [354]:
```
target_store_88 = 88
control_store_88 = composite_score(target_store_88)
```

In [355]:
```
plot_preTrial('totSales',
              df_ready_to_plot_preTrial(target_store_88, control_store_8
8, 'totSales'),
             target_store_88,
              control_store_88)
```



In [356]:
```
plot_preTrial('nCustomers',
              df_ready_to_plot_preTrial(target_store_88, control_store_8
8, 'nCustomers'),
             target_store_88,
             control_store_88)
```



Assessment of trial

The trial period goes from the start of February 2019 to April 2019. We now want to see if there has been an uplift in overall chip sales.

We'll start with scaling the control stores's sales to level simliar to control for any differences between the two stores outside of the trial period.

Once you have selected your control stores, compare each trial and control pair during the trial period. We want to test if total sales are significantly different in the trial period and if so, check if the driver of change is more purchasing customers or more purchases per customers

In [357]:
```
def percenategDiff_with_scaling_factor(feature, target_store, control_st
ore):
    scalingFactor = preTrialMeasures[preTrialMeasures['STORE_NBR'] == ta
rget_store][feature].sum()/preTrialMeasures[preTrialMeasures['STORE_NBR'
] == control_store][feature].sum()
    scaledFeature_control_store = measureOverTimeSales[measureOverTimeSa
les['STORE_NBR'] == control_store][feature].values*scalingFactor
    scaledFeature_target_store = measureOverTimeSales[measureOverTimeSal
es['STORE_NBR'] == target_store][feature].values
    time_series = measureOverTimeSales[measureOverTimeSales['STORE_NBR']
== control_store]['YEARMONTH'].values

    percentageDiff = pd.DataFrame({'YEARMONTH': time_series,
                                   'scaledFeature_target_store': s
caledFeature_target_store,
                                   'scaledFeature_control_store':
scaledFeature_control_store,
                                   'percentageDiff': abs(scaledFea
ture_control_store - scaledFeature_target_store)/scaledFeature_control_s
tore})
    return percentageDiff


def calculate_t_value(df):
    degreeOfFreedom = len(df[df['YEARMONTH'] < '2019-02-01']['percentage
Diff']) - 1
    stdDev = stats.tstd(df[df['YEARMONTH'] < '2019-02-01']['percentageDi
ff'].values)
    t_test_table = pd.DataFrame({'TransactionMonth': df[(df['YEARMONTH']
> '2019-01-01') & (df['YEARMONTH'] < '2019-05-01')]['YEARMONTH'],
                                  'tValue': df[(df['YEARMONTH'] >
'2019-01-01') & (df['YEARMONTH'] < '2019-05-01')]['percentageDiff'].valu
es/stdDev,
                              'Critial_Value': stats.t(df = degreeOfFr
eedom).ppf(0.975)})

    return stdDev, t_test_table



def df_plot_ready_feature(target_store, control_store, feature, stdDev,
df = measureOverTimeSales):

    pastSales = measureOverTimeSales[measureOverTimeSales['STORE_NBR'].i
sin([target_store, control_store])]
    Control95 = pastSales[pastSales['STORE_NBR'] == control_store][featu
re].values*(1+stdDev*2)
    Control5 =  pastSales[pastSales['STORE_NBR'] == control_store][featu
re].values*(1-stdDev*2)
    Trial = pastSales[pastSales['STORE_NBR'] == target_store][feature].v
alues
    Control = pastSales[pastSales['STORE_NBR'] == control_store][feature
].values

    all_feature = pd.DataFrame({'YEARMONTH': pastSales['YEARMONTH'].uniq
ue(),
```

```
              'Control95': Control95,
              'Control5': Control5,
              'Trial': Trial,
              'Control': Control})
    plot_ready_all_feature = pd.melt(all_feature,
        id_vars = ['YEARMONTH'],
        value_vars = ['Control95', 'Control5', 'Trial', 'Control'],
       var_name = 'store_type',
       value_name = feature)
    return df_plot_ready_all_feature

def plot_overtime_between_control_trial(feature, df, target_store, contr
ol_store):
    plt.figure(figsize = (15, 10))
    ax = sns.lineplot(data = df, x = 'YEARMONTH', y = feature, hue = 'st
ore_type');
    plt.ylim(0)
    ymin, ymax = plt.ylim()
    height = ymax - ymin
    interval = plt.xticks()[0][4] - plt.xticks()[0][3]
    ax.set_xticks(df['YEARMONTH'].unique())
    ax.add_patch(plt.Rectangle((plt.xticks()[0][8] - interval/2, ymin),
                         interval, height,
                         fill = True,
                         alpha = 0.3))
    ax.set_title('{} between Trial Store {} and Control Store {}'.format
(feature, target_store, control_store),
                 fontsize = 16)
    ax.lines[0].set_linestyle(':')
    ax.lines[1].set_linestyle(':')
    plt.legend(title = 'Store Type',
               loc='lower left',
               labels = ['95% CI', '5% CI', 'Trial Store '+str(target_st
ore), 'Control Store '+str(control_store)])
    plt.xticks(rotation = 45)
    sns.despine();
```

Store 77 VS Store 233

We can observe that 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. Let's create a more visual version of the trial stores and 95th percentile value of sales of the control store.

The results show that the trial in store 77 is significant different to its control store in the trial period as the trial store performance lies outside the 5% to 95% confidence interval of the control store in two of the three trial months. Let's have a look at assessing this for number of customers as well.

The total number of customers in the trial period for the trial store is signifantly higher than the control store for two out of three months, which indicates a positive trial effect.

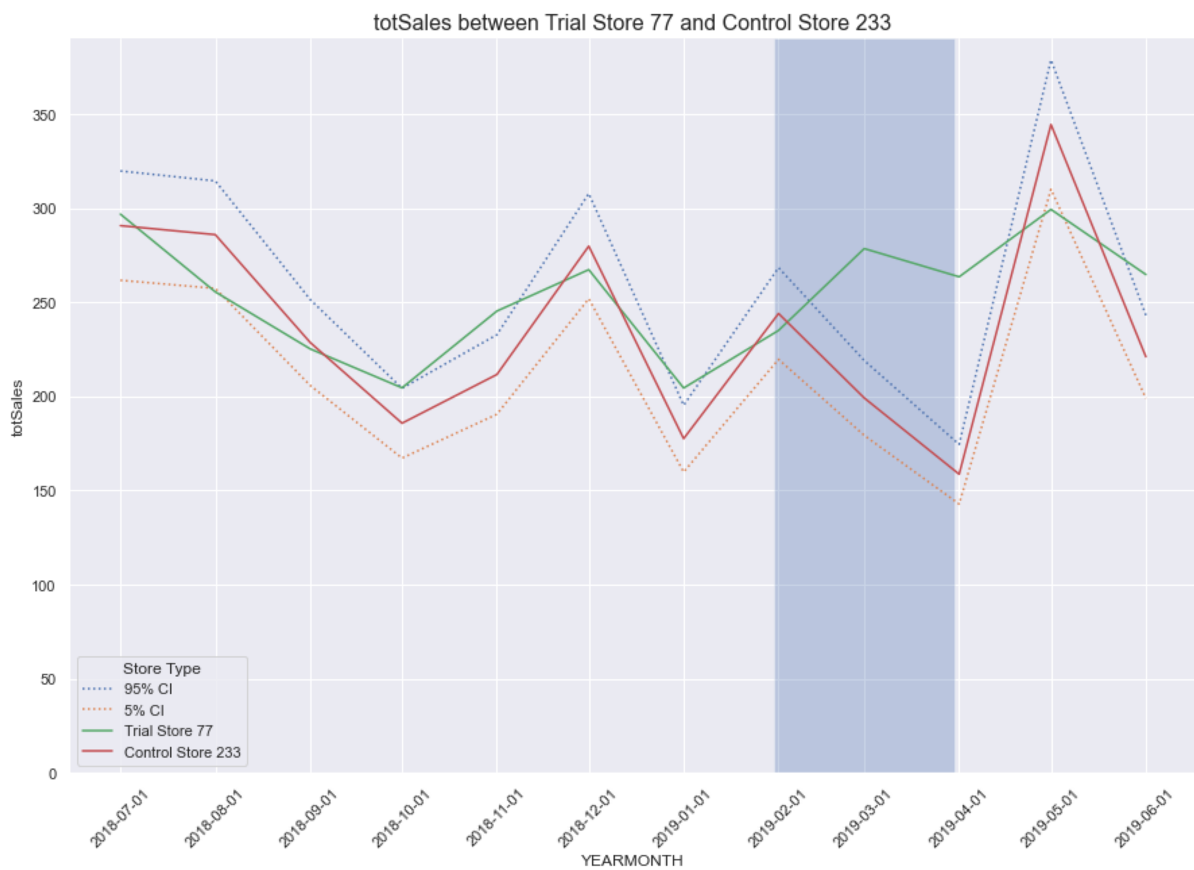
In [358]:
```
stdDev, t_test_table = calculate_t_value(percenategDiff_with_scaling_fac
tor('totSales', target_store_77, control_store_77))
print(stdDev)
t_test_table
```

```
0.049940762641425544
```

Out[358]:

	TransactionMonth	tValue	Critial_Value
7	2019-02-01	1.183534	2.446912
8	2019-03-01	7.339116	2.446912
9	2019-04-01	12.476373	2.446912

In [359]:
```
df_plot_ready_all_totSales_77 = plot_ready_feature(target_store_77, cont
rol_store_77, 'totSales', stdDev)
plot_overtime_between_control_trial('totSales', df_plot_ready_all_totSal
es_77, target_store_77, control_store_77)
```



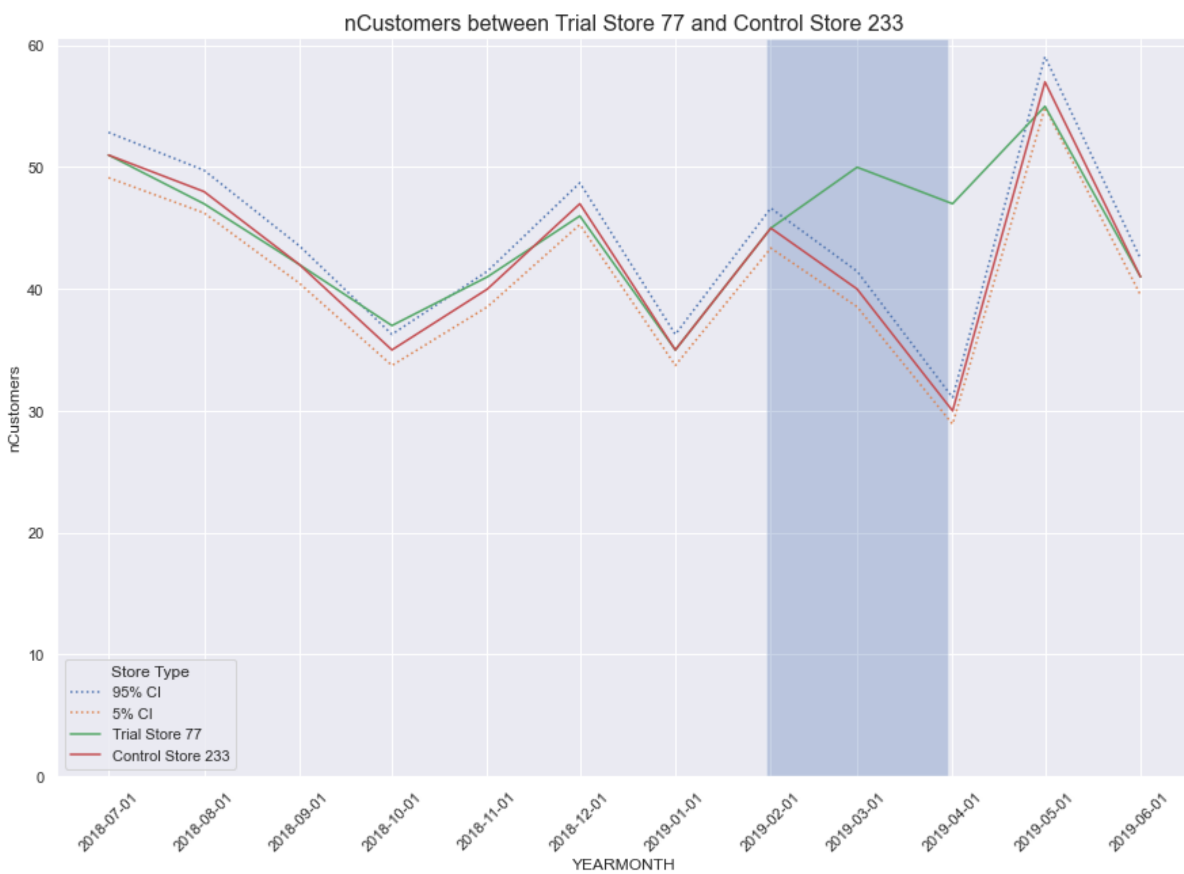
In [360]:
```python
stdDev, t_test_table = calculate_t_value(percenategDiff_with_scaling_factor('nCustomers', target_store_77, control_store_77))
print(stdDev)
t_test_table
```

0.01824074855824395

Out[360]:

	TransactionMonth	tValue	Critial_Value
7	2019-02-01	0.183352	2.446912
8	2019-03-01	13.476388	2.446912
9	2019-04-01	30.778725	2.446912

In [361]:
```python
stdDev, t_test_table = calculate_t_value(percenategDiff_with_scaling_factor('nCustomers', target_store_77, control_store_77))
df_plot_ready_all_nCustomers_77 = plot_ready_feature(target_store_77, control_store_77, 'nCustomers', stdDev)
plot_overtime_between_control_trial('nCustomers', df_plot_ready_all_nCustomers_77, target_store_77, control_store_77)
```



Store 86 VS Store 155

We can observe that t-value is much larger than the Critical Value for 95th percentile value of the t-distribution on March - i.e the increase in sales in the trial store in March is statistically greater than in the control store. But two other months Feb and April during th trial period are not significantly different than the control store.

The results show that the trial in not significantly different to its control store in the trial period as the trial store performance lies inside the 5% to 95% confidence interval of the interval of the control store in two of the three trial months. let's have a look at assessing this for number of customers as well.

It looks like the number of customers is significatly higher in all of three months. This seems to suggest that the trial had a significant impact on increasing the number of customers in trial store 86 but as we saw, sales were not significantly higher. We should check with Category Manager if there were special deals in the trial store that were may have resulted in lower prices, impacting the results.

In [362]:

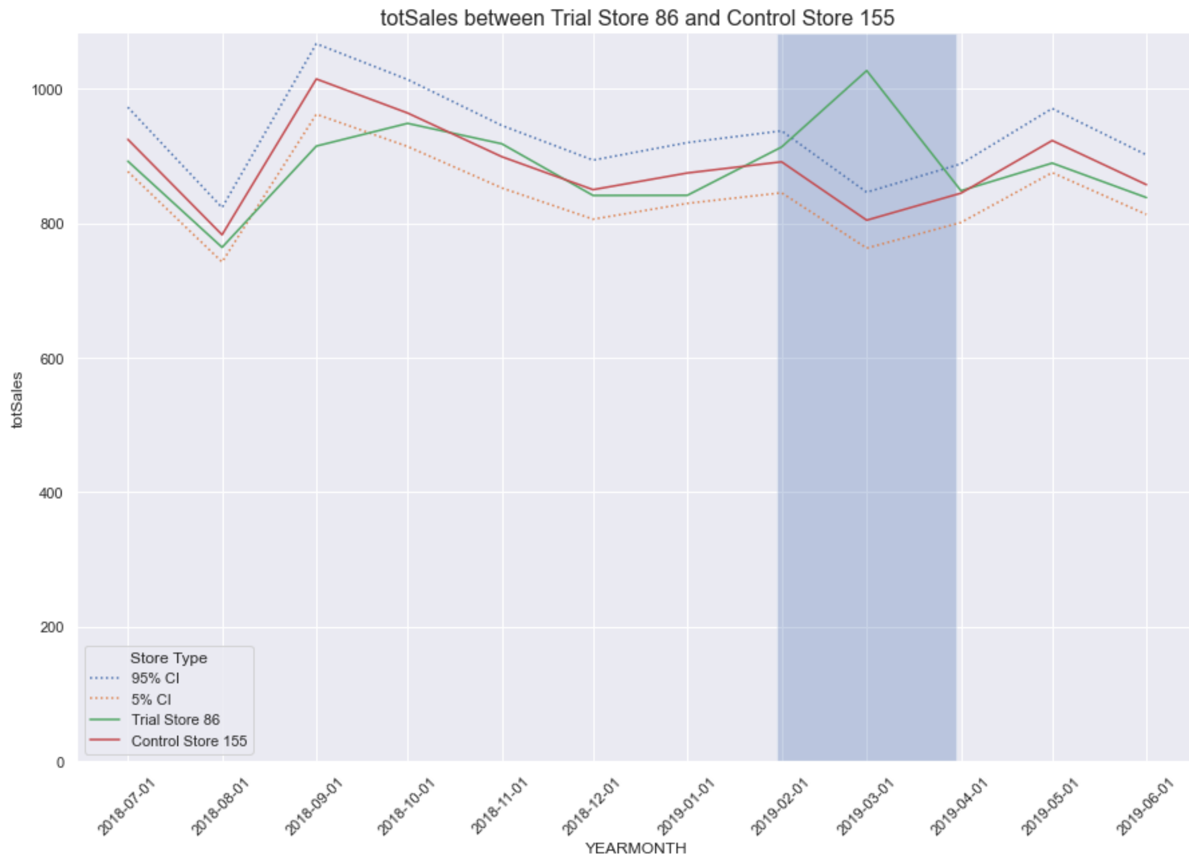
```
stdDev, t_test_table = calculate_t_value(percenategDiff_with_scaling_fac
tor('totSales', target_store_86, control_store_86))
print(stdDev)
t_test_table
```

```
0.025833952854772586
```

Out[362]:

	TransactionMonth	tValue	Critial_Value
7	2019-02-01	2.179542	2.446912
8	2019-03-01	12.226922	2.446912
9	2019-04-01	1.364580	2.446912

In [363]:
```
df_plot_ready_all_totSales_86 = plot_ready_feature(target_store_86, control_store_86, 'totSales', stdDev)
plot_overtime_between_control_trial('totSales', df_plot_ready_all_totSales_86, target_store_86, control_store_86)
```



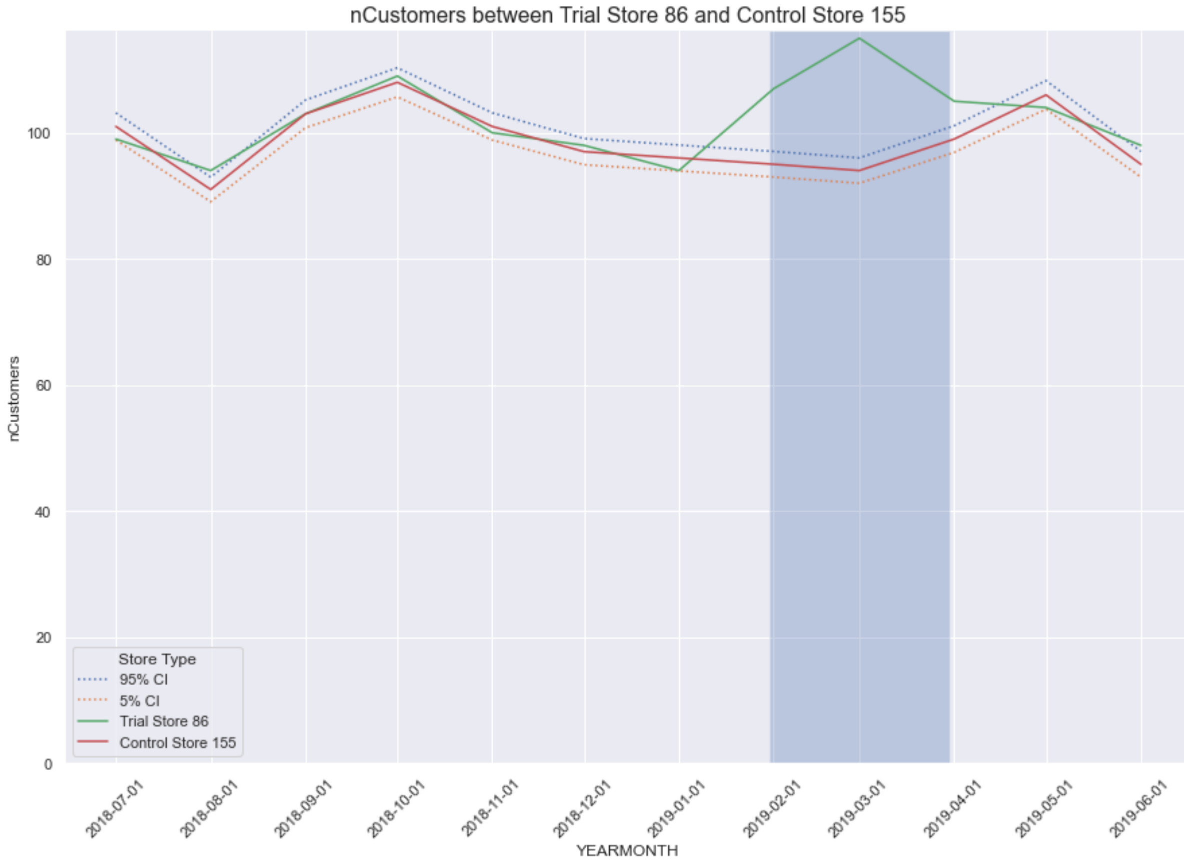
In [364]:
```
stdDev, t_test_table = calculate_t_value(percenategDiff_with_scaling_factor('nCustomers', target_store_86, control_store_86))
print(stdDev)
t_test_table
```

0.010687444701395238

Out[364]:

	TransactionMonth	tValue	Critial_Value
7	2019-02-01	11.819082	2.446912
8	2019-03-01	20.903430	2.446912
9	2019-04-01	5.670772	2.446912

In [365]:
```
df_plot_ready_all_nCustomers_86 = plot_ready_feature(target_store_86, control_store_86, 'nCustomers', stdDev)
plot_overtime_between_control_trial('nCustomers', df_plot_ready_all_nCustomers_86, target_store_86, control_store_86)
```



Store 88 VS Store 237

We've now found store 237 to be a suitable control store for trial store 88. Again, let's check visually if the drivers are indeed similar in the period before the trial. We'look at sales first.

The results show that the trial store 88 is significantly different than the control store 237 in the trial pperiod as the trial store lies outside of the 5% to 95% confidence interval of the control store in two of the three trial months. Let's have a look at assessing this for number of customers as well.

The total number of customers in the trial period for the trial store is signifantly higher than the control store for two out of three months, which indicates a positive trial effect.

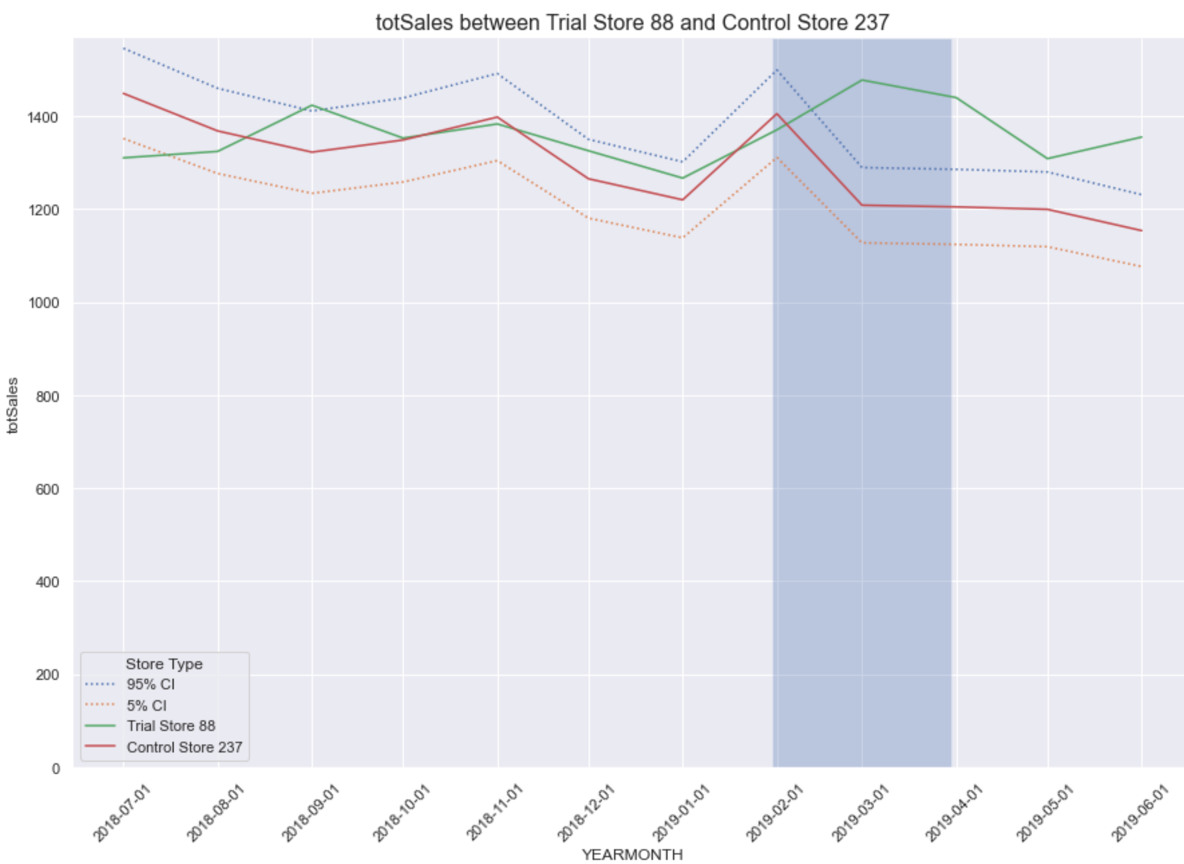
In [366]:
```
stdDev, t_test_table = calculate_t_value(percenategDiff_with_scaling_factor('totSales', target_store_88, control_store_88))
print(stdDev)
t_test_table
```

```
0.033467867303078889
```

Out[366]:

	TransactionMonth	tValue	Critial_Value
7	2019-02-01	0.781270	2.446912
8	2019-03-01	6.595668	2.446912
9	2019-04-01	5.768527	2.446912

In [367]:
```
df_plot_ready_all_totSales_88 = plot_ready_feature(target_store_88, control_store_88, 'totSales', stdDev)
plot_overtime_between_control_trial('totSales', df_plot_ready_all_totSales_88, target_store_88, control_store_88)
```



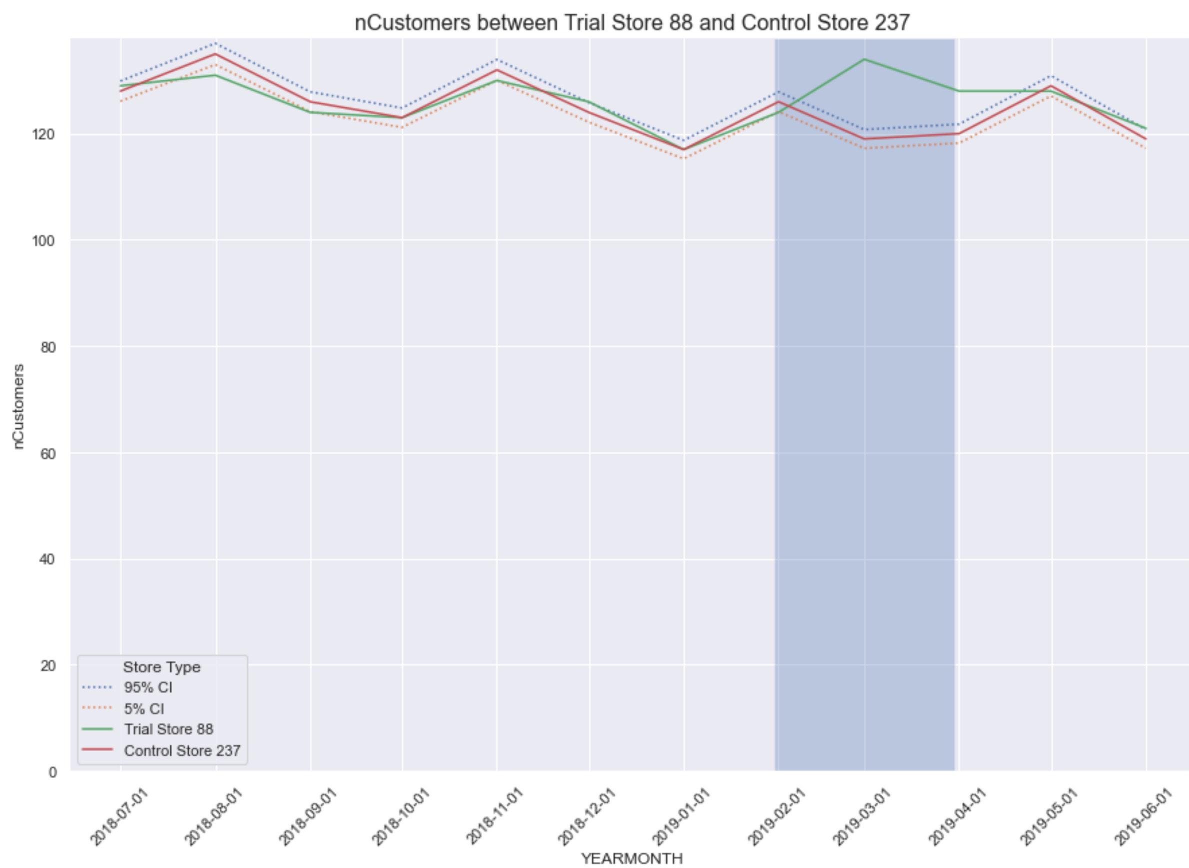
In [368]:
```
stdDev, t_test_table = calculate_t_value(percenategDiff_with_scaling_factor('nCustomers', target_store_88, control_store_88))
print(stdDev)
t_test_table
```

0.00741024435207507

Out[368]:

	TransactionMonth	tValue	Critial_Value
7	2019-02-01	1.387456	2.446912
8	2019-03-01	17.873693	2.446912
9	2019-04-01	9.814423	2.446912

In [369]:
```
df_plot_ready_all_nCustomers_88 = plot_ready_feature(target_store_88, control_store_88, 'nCustomers', stdDev)
plot_overtime_between_control_trial('nCustomers', df_plot_ready_all_nCustomers_88, target_store_88, control_store_88)
```



Summary

We've found control store 233, 155, 237 for trial store 77, 86, 88 respectively. The results for trial stores 77 and 88 during the trial period show a significant difference in the at least two of the three trial months but this is not the case for trial store 86. We can check with the client if the implementation of the trial was different in trial store 86 but overall, The trial shows a significant increase in sales.