Quantium Virtual Internship - Retail Strategy and Analytics - Task 2

In [2]:
```
# Loading required libraries
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
import datetime
from scipy import stats
```

In [12]:
```
#   Ignoring any warnings.

import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)
```

In [3]:
```
# Read data files into data frames
df = pd.read_csv('QVI_data.csv')
df
```

Out[3]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	
3	1003	2019-03-08	1	4	106	Natural ChipCo Hony Soy Chckn175g	
4	1004	2018-11-02	1	5	96	WW Original Stacked Chips 160g	
...	...	...	...	...	...	...	
264829	2370701	2018-12-08	88	240378	24	Grain Waves Sweet Chilli 210g	
264830	2370751	2018-10-01	88	240394	60	Kettle Tortilla ChpsFeta&Garlic 150g	
264831	2370961	2018-10-24	88	240480	70	Tyrrells Crisps Lightly Salted 165g	
264832	2370961	2018-10-27	88	240481	65	Old El Paso Salsa Dip Chnky Tom Ht300g	
264833	2373711	2018-12-14	88	241815	16	Smiths Crinkle Chips Salt & Vinegar 330g	

264834 rows × 12 columns

The client has selected store numbers 77, 86 and 88 as trial stores with a trial period of Feb 2019 to April 2019. The client also wants control stores to be established stores that are operational for the entire observation period.

We would want to match trial 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 Monthly number of transactions per customer To choose the control stores, we will create the metrics of interest and filter to stores that are present throughout the pre-trial period.

First, we want to add a column with the year/month of the transaction.

In [4]:
```
# Change DATE column to store dates as datetimes
df['DATE'] = pd.to_datetime(df['DATE'])

# Then add a YEARMONTH column
df['YEARMONTH'] = df['DATE'].dt.strftime('%Y%m').astype('int64')
df
```

Out[4]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	
3	1003	2019-03-08	1	4	106	Natural ChipCo Hony Soy Chckn175g	
4	1004	2018-11-02	1	5	96	WW Original Stacked Chips 160g	
...	...	...	...	...	...	...	
264829	2370701	2018-12-08	88	240378	24	Grain Waves Sweet Chilli 210g	
264830	2370751	2018-10-01	88	240394	60	Kettle Tortilla ChpsFeta&Garlic 150g	
264831	2370961	2018-10-24	88	240480	70	Tyrrells Crisps Lightly Salted 165g	
264832	2370961	2018-10-27	88	240481	65	Old El Paso Salsa Dip Chnky Tom Ht300g	
264833	2373711	2018-12-14	88	241815	16	Smiths Crinkle Chips Salt & Vinegar 330g	

264834 rows × 13 columns

Next, we want to create a function that will be able to calculate the total sales, number of customers, transactions per customer, chips per customer and the average price per unit for each store and month.

In [5]:

```python
# Define the metrics and calculate them
grouped_df = df.groupby(["STORE_NBR","YEARMONTH"])
tot_sales = grouped_df.TOT_SALES.sum()
n_cust = grouped_df.LYLTY_CARD_NBR.nunique()
ntrans_percust = grouped_df.TXN_ID.size()/n_cust
```

```
nchips_pertrans = grouped_df.PROD_QTY.sum()/grouped_df.TXN_ID.size()
avg_priceperunit = tot_sales/grouped_df.PROD_QTY.sum()
# Put the metrics together in an array
metric_arrays =  [tot_sales, n_cust, ntrans_percust, nchips_pertrans, avg_priceperu

# Create the metrics table fro mthe array
metrics_df = pd.concat(metric_arrays, axis=1)
# Give the columns labels
metrics_df.columns = ['tot_sales', 'n_cust', 'ntrans_percust', 'nchips_pertrans', '
metrics_df = metrics_df.reset_index()
```

In [6]:
```
# Filter to select the stores with full observation periods
month_counts = metrics_df.groupby('STORE_NBR').YEARMONTH.nunique().reset_index()
stores_fullobs = month_counts[month_counts.YEARMONTH ==12].STORE_NBR
pretrial_metrics = metrics_df[metrics_df['STORE_NBR'].isin(stores_fullobs)]

# Then filter to keep only the pre-trial period data
pretrial_metrics = pretrial_metrics.loc[pretrial_metrics.YEARMONTH < 201902]
pretrial_metrics
```

Out[6]:

	STORE_NBR	YEARMONTH	tot_sales	n_cust	ntrans_percust	nchips_pertrans	avg_pr
0	1	201807	206.9	49	1.061224	1.192308	
1	1	201808	176.1	42	1.023810	1.255814	
2	1	201809	278.8	59	1.050847	1.209677	
3	1	201810	188.1	44	1.022727	1.288889	
4	1	201811	192.6	46	1.021739	1.212766	
...	...	...	...	...	...	...	
3159	272	201809	304.7	32	1.125000	1.972222	
3160	272	201810	430.6	44	1.159091	1.941176	
3161	272	201811	376.2	41	1.097561	1.933333	
3162	272	201812	403.9	47	1.000000	1.893617	
3163	272	201901	423.0	46	1.086957	1.920000	

1820 rows × 7 columns

Now we need to work out a way of ranking how similar each potential control store is to the trial store. We can calculate how correlated the performance of each potential control store is to the trial store.

In [15]:
```
def calc_corr(trial, metric_col, input_table = pretrial_metrics):
    trial_stores = [77, 86, 88]
    control_stores = stores_fullobs[~stores_fullobs.isin(trial_stores)] # all store
    # Keep the trial store values to perform correlation with
    trial_vals = input_table[input_table["STORE_NBR"] == trial][metric_col].reset_i
    corr_table = pd.DataFrame(columns = ['YEARMONTH', 'trial_store', 'control_store
    # Find the correlation for each control store
    for control in control_stores:
        # Keep the control store values to perform correlation with
        control_vals = input_table[input_table["STORE_NBR"] == control][metric_col]
        corr_row = pd.DataFrame(columns = ['YEARMONTH', 'trial_store', 'control_sto
        corr_row.YEARMONTH = list(input_table.loc[input_table.STORE_NBR == control]
        corr_row.trial_store = trial
        corr_row.control_store = control
        corr_row.correlation = control_vals.corrwith(trial_vals, axis=1)
        corr_table = pd.concat([corr_table, corr_row]) # add each store's block to
    return (corr_table)
```

In [16]:
```
trial_stores = [77, 86, 88]
corr_sections = []
for store in trial_stores:
    corr_section = calc_corr(store, ['tot_sales', 'n_cust', 'ntrans_percust', 'nchi
    if not corr_section.empty and not corr_section.isna().all().all():
        corr_sections.append(corr_section)
corr_table = pd.concat(corr_sections, ignore_index=True)
```

In [17]:
```
corr_table
```

Out[17]:

	YEARMONTH	trial_store	control_store	correlation
0	201807	77	1	0.070544
1	201808	77	1	0.027332
2	201809	77	1	0.002472
3	201810	77	1	-0.019991
4	201811	77	1	0.030094
...	...	...	...	...
5392	201809	88	272	0.533160
5393	201810	88	272	0.591056
5394	201811	88	272	0.566378
5395	201812	88	272	0.594442
5396	201901	88	272	0.621775

5397 rows × 4 columns

Apart from correlation, we can also calculate a standardised metric based on the absolute difference between the trial store's performance and each control store's performance. Write a function to calculate the magnitude distance.

In [18]:
```
def calc_magdist(trial, metric_col, input_table = pretrial_metrics):
    trial_stores = [77, 86, 88]
    control_stores = stores_fullobs[~stores_fullobs.isin(trial_stores)] # all store
    dist_table = pd.DataFrame() # to store the distances for each store and month
    for control in control_stores: # calculate for each control store
        dist_row = pd.DataFrame()
        # Calculate the distance as an absolute value
        dist_row = abs(input_table[input_table["STORE_NBR"] == trial].reset_index()
                        - input_table[input_table["STORE_NBR"] == control].reset_in
        dist_row.insert(0,'YEARMONTH', list(input_table.loc[input_table.STORE_NBR =
        dist_row.insert(1,'trial_store', trial)
        dist_row.insert(2,'control_store', control)
        dist_table = pd.concat([dist_table, dist_row])

    for col in metric_col: # then loop over each column to find the max and min dis
        maxdist = dist_table[col].max()
        mindist = dist_table[col].min()
        dist_table[col] = 1-(dist_table[col] - mindist)/(maxdist-mindist) # normali
        # also give an average magnitude over all metrics per month and store pair
    dist_table['mag_measure'] = dist_table[metric_col].mean(axis=1)
    return (dist_table)
```

Now we will use the functions to find the control stores! We'll select control stores based on how similar monthly total sales in dollar amounts and monthly number of customers are to the trial stores. So we will need to use our functions to get four scores, two for each of total sales and total customers.

In [19]:
```
def calc_corrdist_score (trial, metric_col, input_table=pretrial_metrics):
    # Calculate the correlations and magnitudes for all months
    corr_vals = calc_corr(trial, metric_col, input_table)
    mag_vals = calc_magdist(trial, metric_col, input_table)
    mag_vals = mag_vals.drop(metric_col, axis=1) # For one metric, the two columns

    # Combine correlations and magnitudes together to one df
    combined_corr_dist = pd.merge(corr_vals, mag_vals, on=["YEARMONTH", "trial_stor

    # Average correlations and distances over the pre-trial months
    avg_corrmag = combined_corr_dist.groupby(["trial_store", "control_store"]).mean

    # Find a combined score by taking the weighted average of the correlations and
    corr_weight = 0.5
    avg_corrmag['combined_score'] = corr_weight*avg_corrmag['correlation'] + (1-cor

    return(avg_corrmag)
```

In [22]:
```
def find_highestscore(trial):
    # Obtain the scores for the tot_sales and n_cust metrics separately
    scores_tot_sales = calc_corrdist_score (trial, ['tot_sales'])
    scores_n_cust = calc_corrdist_score (trial, ['n_cust'])
    # Create a data table to store the composite results in - stores are also
    scores_control = pd.DataFrame()
    scores_control['control_store'] = scores_tot_sales.control_store
    # Calculate the composite scores
    scores_control['correlation'] = 0.5*scores_tot_sales.correlation + 0.5*scores_n
    scores_control['mag_measure'] = 0.5*scores_tot_sales.mag_measure + 0.5*scores_n
    scores_control['scores'] = 0.5*scores_tot_sales.combined_score + 0.5*scores_n_c
    return(scores_control.sort_values(by = 'scores', ascending = False).reset_index
```

In [23]:
```
# Now find the control stores with the highest scores for each of the trial stores
trial_stores = [77, 86, 88]
for trial in trial_stores:
    print('Trial store: ', trial)
    print(find_highestscore(trial))
    print()
```

```
Trial store:  77
   control_store  correlation  mag_measure    scores
0            233          1.0     0.989804  0.994902
1             41          1.0     0.972041  0.986020
2             46          1.0     0.969523  0.984762
3             53          1.0     0.968421  0.984211
4            111          1.0     0.967981  0.983991

Trial store:  86
   control_store  correlation  mag_measure    scores
0            155          1.0     0.976324  0.988162
1            109          1.0     0.968180  0.984090
2            225          1.0     0.965044  0.982522
3            229          1.0     0.957995  0.978997
4            101          1.0     0.945394  0.972697

Trial store:  88
   control_store  correlation  mag_measure    scores
0             40          1.0     0.941789  0.970895
1             26          1.0     0.917859  0.958929
2             72          1.0     0.908157  0.954079
3             58          1.0     0.900435  0.950217
4             81          1.0     0.887572  0.943786
```

Insights from Store Pairing

From the output, we observe the following:

- Store 233 has the highest score for trial store 77
- Store 155 has the highest score for trial store 86
- Store 40 has the highest score for trial store 88

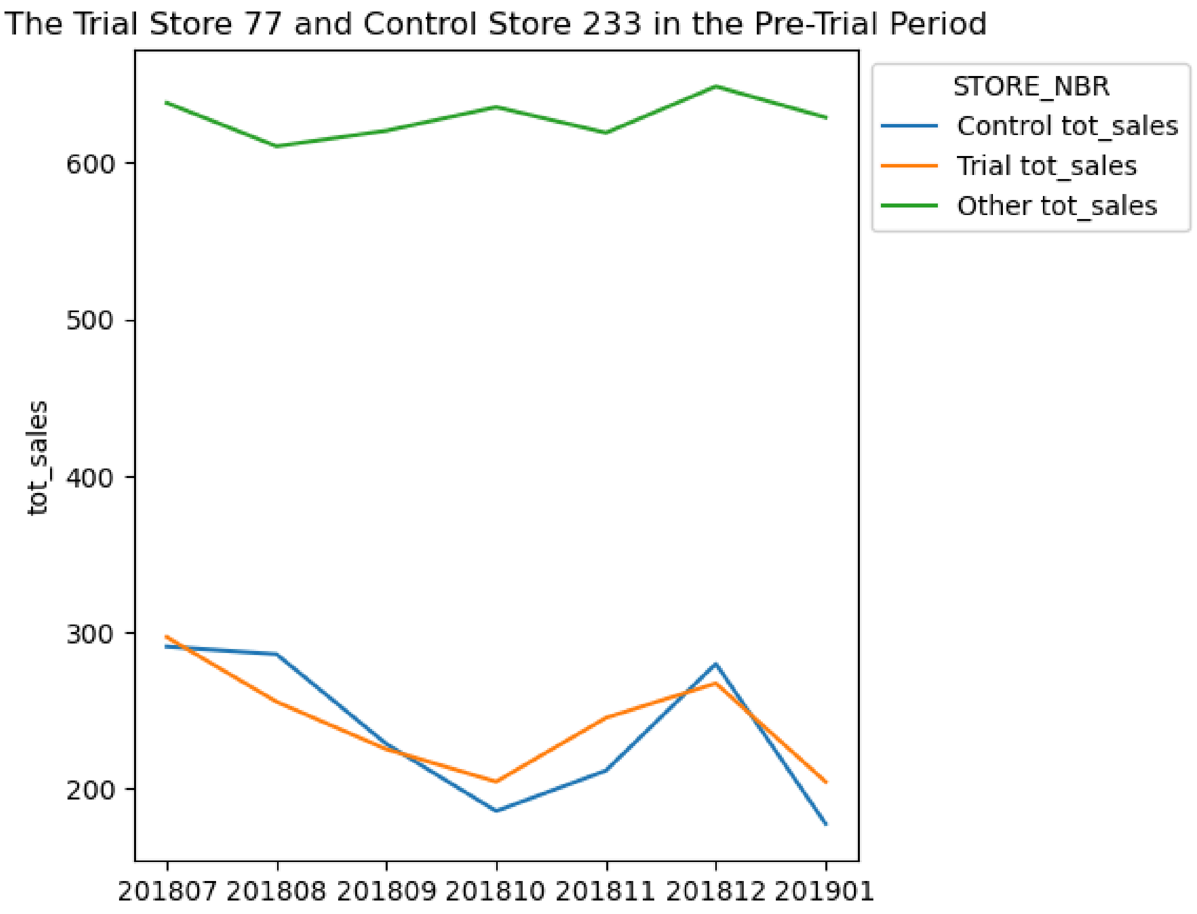
Note that the combined store scores for the control cases of trial store 88 are lower than those of stores 77 and 86. This may suggest that the control stores may not match store 88 as well as they do for the other trial stores.

Next Steps

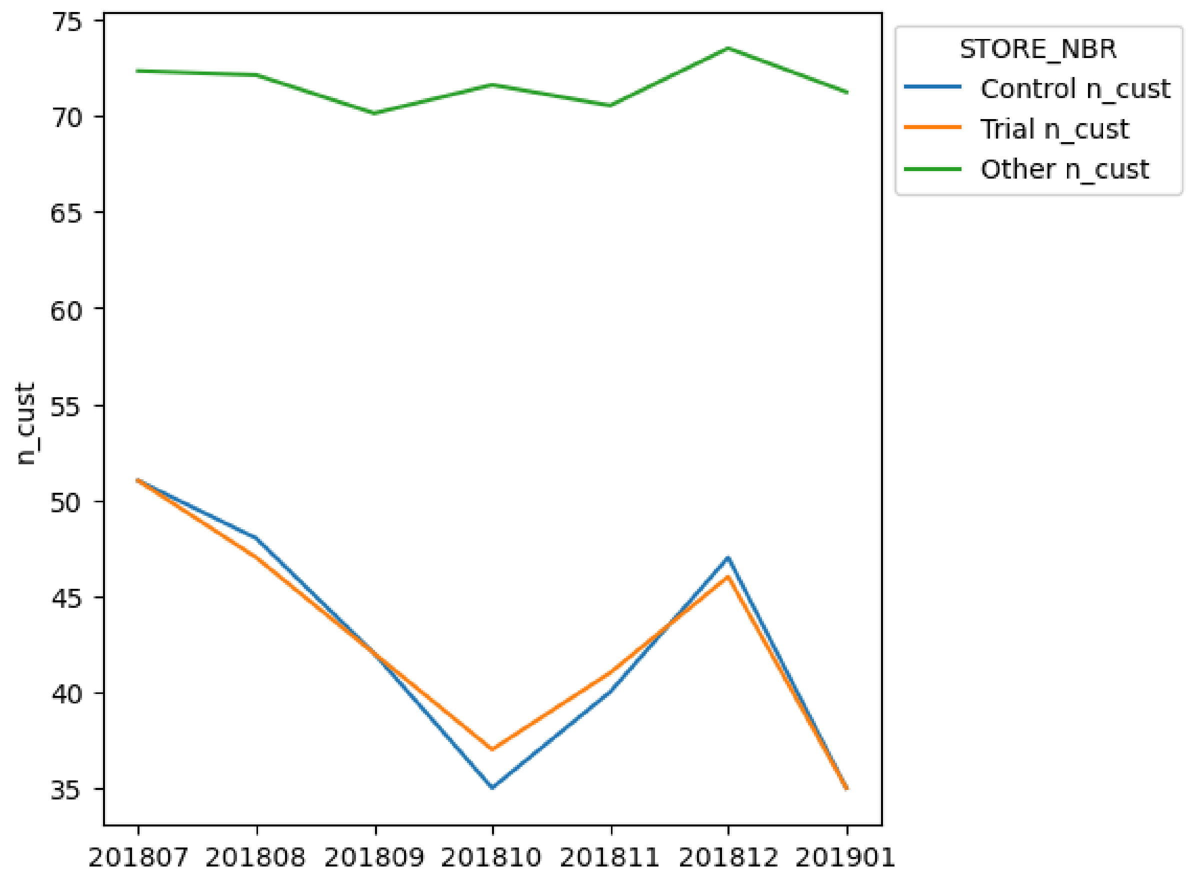
Now that we have identified the control stores, we can visually inspect if the drivers are similar between these stores and the trial stores in the pre-trial period.

In [26]:
```
def make_plots(storepair, metric_col):
    trial = storepair[0]
    control = storepair[1]
    trial_plot = pretrial_metrics[pretrial_metrics.STORE_NBR == trial][['YEARMONTH'
    trial_plot = trial_plot.rename(columns = {metric_col: metric_col+'_trial'})
    control_plot = pretrial_metrics[pretrial_metrics.STORE_NBR == control][['YEARMO
    control_plot = control_plot.rename(columns = {metric_col: metric_col+'_control'

    other_stores = pretrial_metrics.loc[(pretrial_metrics.STORE_NBR != 77)][['YEARM
    other_stores = other_stores.loc[(pretrial_metrics.STORE_NBR != 233)]
    plot_other = other_stores.groupby('YEARMONTH')[metric_col].mean()

    ax = control_plot.plot.line(x = "YEARMONTH", y = metric_col+'_control', use_ind
    ax_trial = trial_plot.plot.line(x = "YEARMONTH", y = metric_col+'_trial', use_i
    ax_other = plot_other.plot.line(use_index = False, ax=ax, label = 'Other '+ met
    ax.set_ylabel(metric_col)
    plt.legend(title = 'STORE_NBR', loc = "upper left",bbox_to_anchor=(1.0, 1.0))
    positions = (0,1,2,3,4,5,6)
    labels = ("201807", '201808', '201809', '201810', '201811', '201812', '201901')
    plt.xticks (positions, labels)
    titlestr = 'The Trial Store ' + str(storepair[0]) + ' and Control Store ' + str
    plt.tight_layout()
    ax.set_title(titlestr)
```

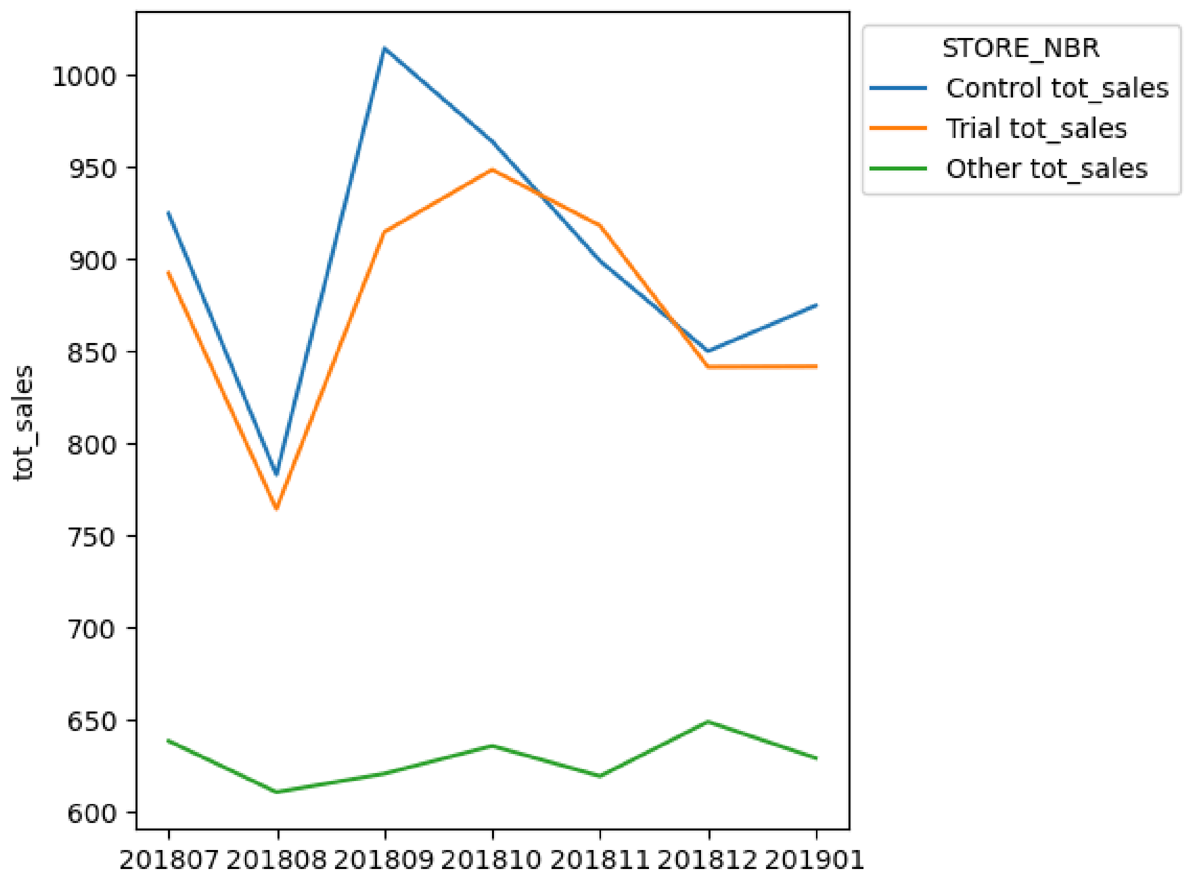
```
In [27]: storepair = [[77, 233], [86, 155], [88, 40]]
         metric_col = ['tot_sales', 'n_cust']
         for pair in storepair:
             for metric in metric_col:
                 make_plots(pair, metric)
```

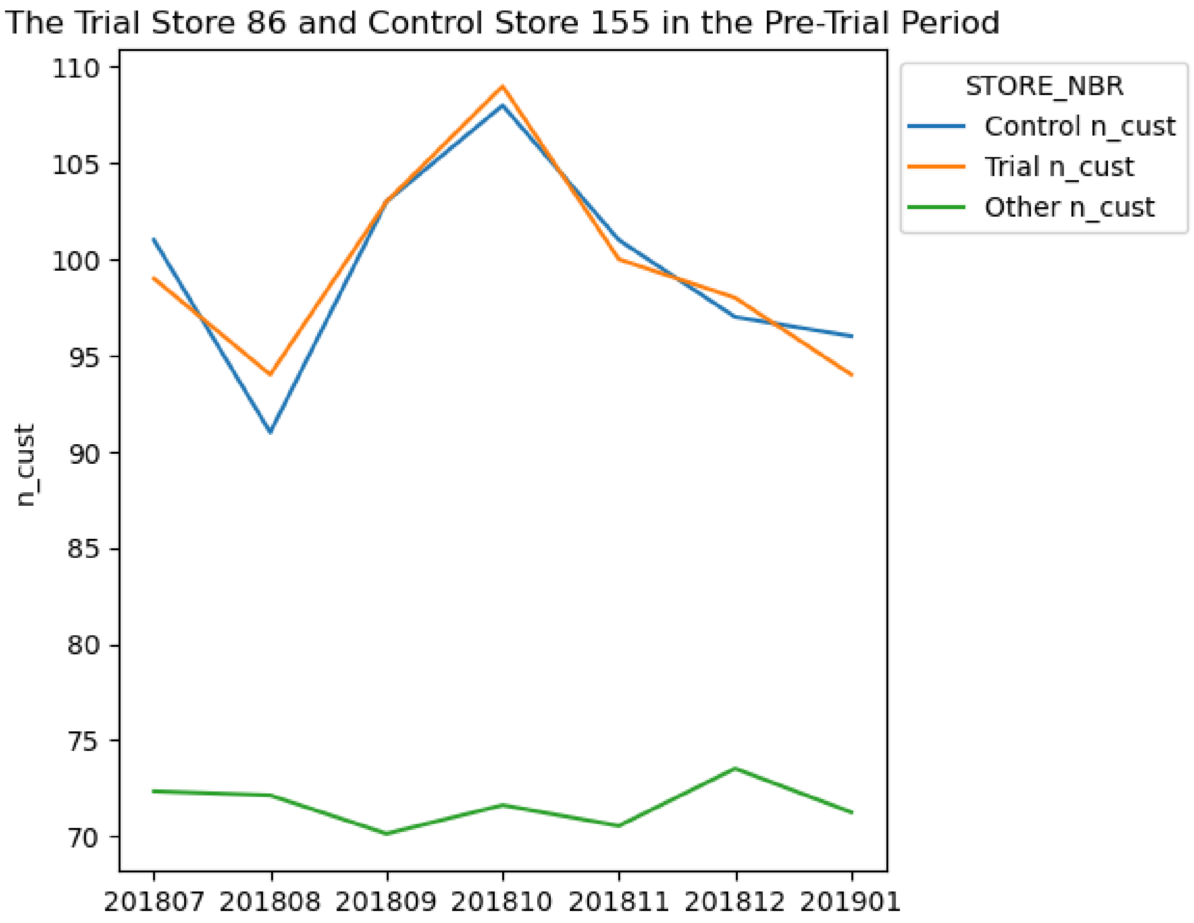


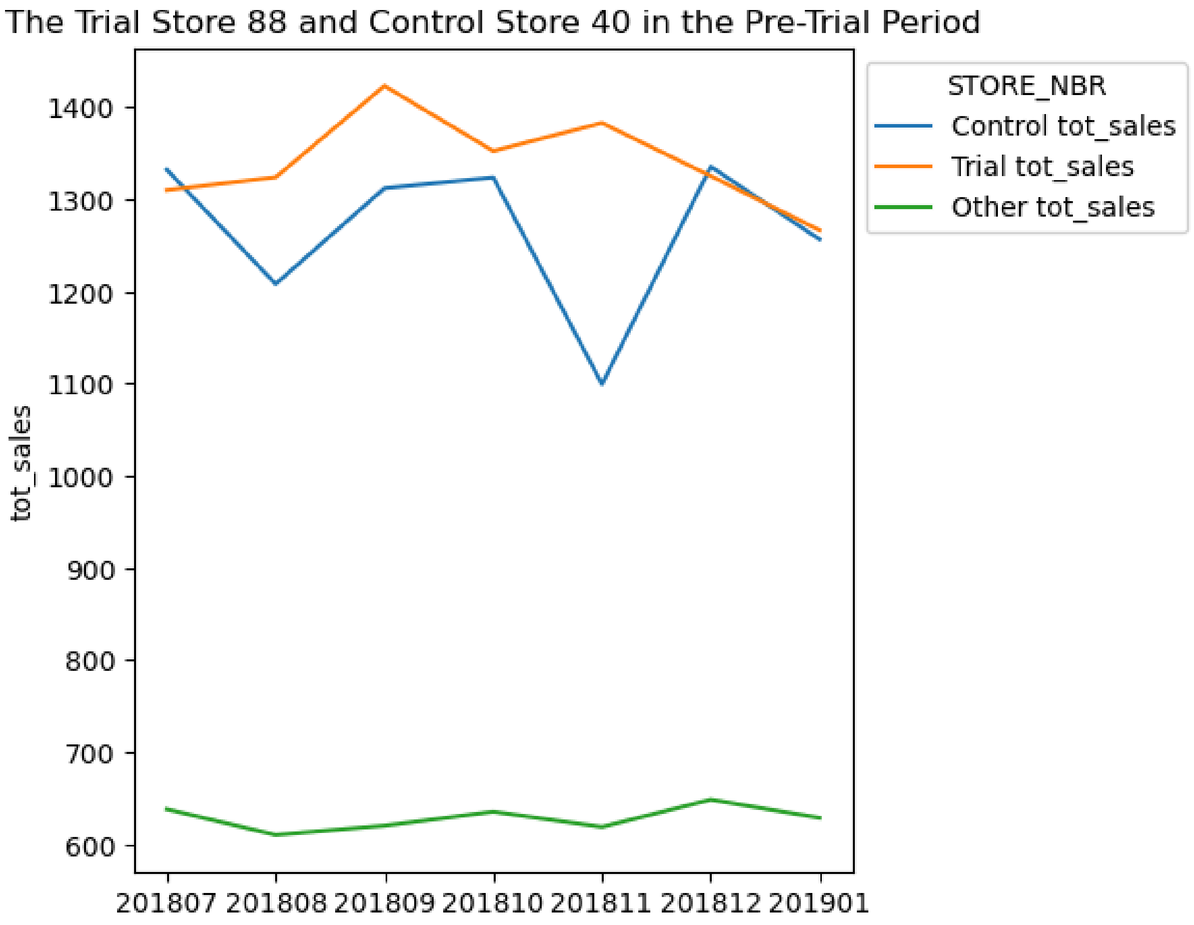
The Trial Store 77 and Control Store 233 in the Pre-Trial Period

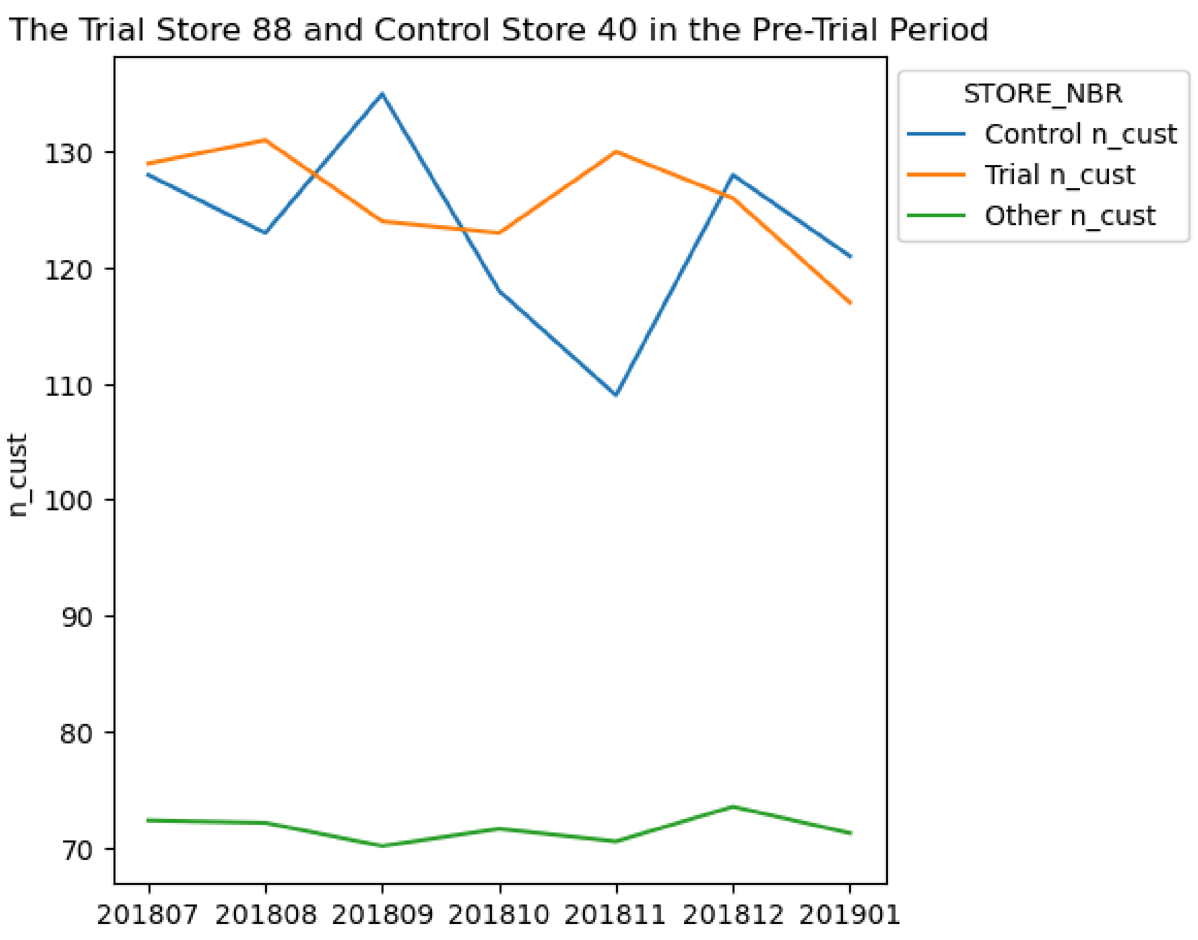


The Trial Store 86 and Control Store 155 in the Pre-Trial Period




The Trial Store 86 and Control Store 155 in the Pre-Trial Period
n_cust
110
105
100
95
90
85
80
75
70
201807
201808
201809
201810
201811
201812
201901
STORE_NBR
Control n_cust
Trial n_cust
Other n_cust


The Trial Store 88 and Control Store 40 in the Pre-Trial Period
tot_sales
1400
1300
1200
1100
1000
900
800
700
600
201807
201808
201809
201810
201811
201812
201901
STORE_NBR
Control tot_sales
Trial tot_sales
Other tot_sales



The metrics of the control and trial stores look reasonably similar in the pre-trial period.

Now, we want to see if there has been an uplift in overall chip sales. We'll start with scaling the control store's sales to a level similar to control for any differences between the two stores outside of the trial period.

In [28]:
```
# Calculate the scaling factor for the store pairs
scale_store77 = pretrial_metrics[pretrial_metrics.STORE_NBR == 77]['tot_sales'].sum
scale_store86 = pretrial_metrics[pretrial_metrics.STORE_NBR == 86]['tot_sales'].sum
scale_store88 = pretrial_metrics[pretrial_metrics.STORE_NBR == 88]['tot_sales'].sum
```

In [29]:
```
# Extract the control store data from the df and scale according to the store
scaled_control233 = metrics_df[metrics_df.STORE_NBR.isin([233])][['STORE_NBR', "YEA
scaled_control233.tot_sales *= scale_store77
scaled_control155 = metrics_df[metrics_df.STORE_NBR.isin([155])][['STORE_NBR', "YEA
scaled_control155.tot_sales *= scale_store86
scaled_control40 = metrics_df[metrics_df.STORE_NBR.isin([40])][['STORE_NBR', "YEARM
scaled_control40.tot_sales *= scale_store88

# Combine the scaled control stores to a single df
scaledsales_control = pd.concat([scaled_control233, scaled_control155, scaled_contr
scaledsales_control = scaledsales_control.rename(columns = {'tot_sales':'scaled_tot
# Get the trial period of scaled control stores
scaledsales_control_trial = scaledsales_control[(scaledsales_control.YEARMONTH>=201
# Get the trial period of the trial stores
trialsales = metrics_df[metrics_df.STORE_NBR.isin([77,86,88])][['STORE_NBR', "YEARM
trialsales = trialsales.rename(columns = {'STORE_NBR': 'TRIAL_NBR'})
trialsales_trial = trialsales[(trialsales.YEARMONTH >= 201902) & (trialsales.YEARMO
```

Now that we have comparable sales figures for the control store, we can calculate the percentage difference between the scaled control sales and the trial store's sales during the trial period.

In [30]:

```
# Calculate the percentage difference between the control and trial store pairs for
percentdiff = scaledsales_control.copy()
percentdiff[['TRIAL_NBR', 'tot_sales_t']] = trialsales[['TRIAL_NBR', 'tot_sales']]
percentdiff = percentdiff.rename(columns = {'scaled_tot_sales' : 'scaled_sales_c'})
percentdiff['sales_percent_diff'] = (percentdiff.tot_sales_t-percentdiff.scaled_sal
                                    /(0.5*((percentdiff.scaled_sales_c+percentdiff.
percentdiff.head()
```

Out[30]:

	CONTROL_NBR	YEARMONTH	scaled_sales_c	TRIAL_NBR	tot_sales_t	sales_percent_diff
0	233	201807	297.565550	77	296.8	-0.002576
1	233	201808	292.652187	77	255.5	-0.135554
2	233	201809	233.998916	77	225.2	-0.038323
3	233	201810	190.085733	77	204.5	0.073060
4	233	201811	216.597421	77	245.3	0.124281

Let's see if the difference is significant using a t-test. Our null hypothesis is that the trial period is the same as the pre-trial period; we will test with a null hypothesis that there is a 0-percent between the trial and control stores.

In [32]:

```
# Calculate standard deviation and mean for each trial store in the pre-trial perio
pretrial_percentdiff = percentdiff[percentdiff.YEARMONTH < 201902]
std_mean = pretrial_percentdiff.groupby('TRIAL_NBR')['sales_percent_diff'].agg(['st

# Iterate over each store pair
for trialstore, controlstore in storepair:
    print(f"Trial store - {trialstore}; control store - {controlstore}")
    print("Month : t-statistic")

    # Get standard deviation and mean for current trial store
    std, mean = std_mean.loc[std_mean.TRIAL_NBR == trialstore, ['std', 'mean']].val

    # Filter trial period data for current trial store
    trialperiod = percentdiff[(percentdiff.YEARMONTH >= 201902) & (percentdiff.YEAR

    # Calculate t-statistic for each month in the trial period
    for month, xval in trialperiod.groupby('YEARMONTH')['sales_percent_diff'].sum()
        tstat = (xval - mean) / std
        print(f"{month} : {tstat}")
    print()
```

```
# Calculate 95th percentile value with 6 degrees of freedom
print('95th percentile value:', stats.t.ppf(1-0.05, 6))
```

```
Trial store - 77; control store - 233
Month : t-statistic
201902 : -0.7171038288055838
201903 : 3.035317928855674
201904 : 4.708944418758219

Trial store - 86; control store - 155
Month : t-statistic
201902 : 1.41336187759216
201903 : 7.1230638460421485
201904 : 0.8863824572944236

Trial store - 88; control store - 40
Month : t-statistic
201902 : -0.5481633746817577
201903 : 1.0089992743637823
201904 : 0.9710006270463672

95th percentile value: 1.9431802803927816
```

Insights from T-Statistics

We observe that the t-value for trial store 77 is significantly higher than the 95th percentile value of the t-distribution for March and April. This indicates that the increase in sales in trial store 77 during these months is statistically significant compared to the control store. Similarly, we see a significant increase in sales for trial store 86 in March.

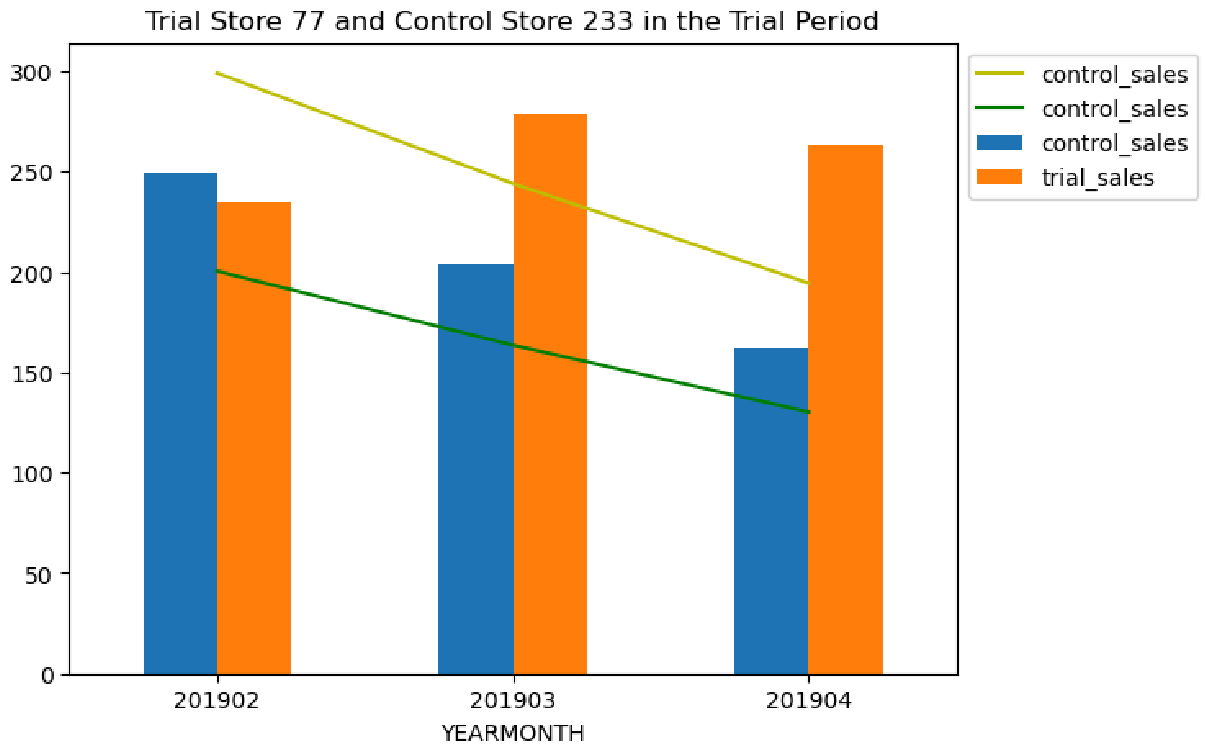
Visualizing Sales Data

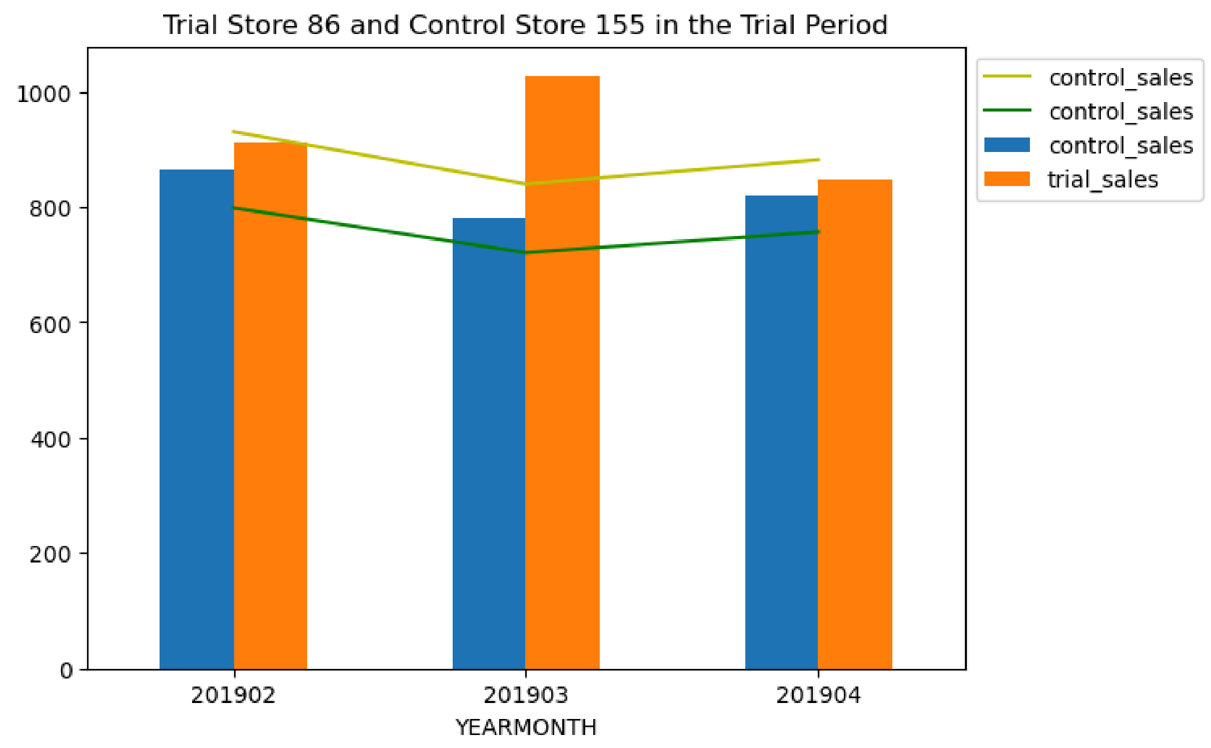
Let's visualize the sales data to better understand these insights:

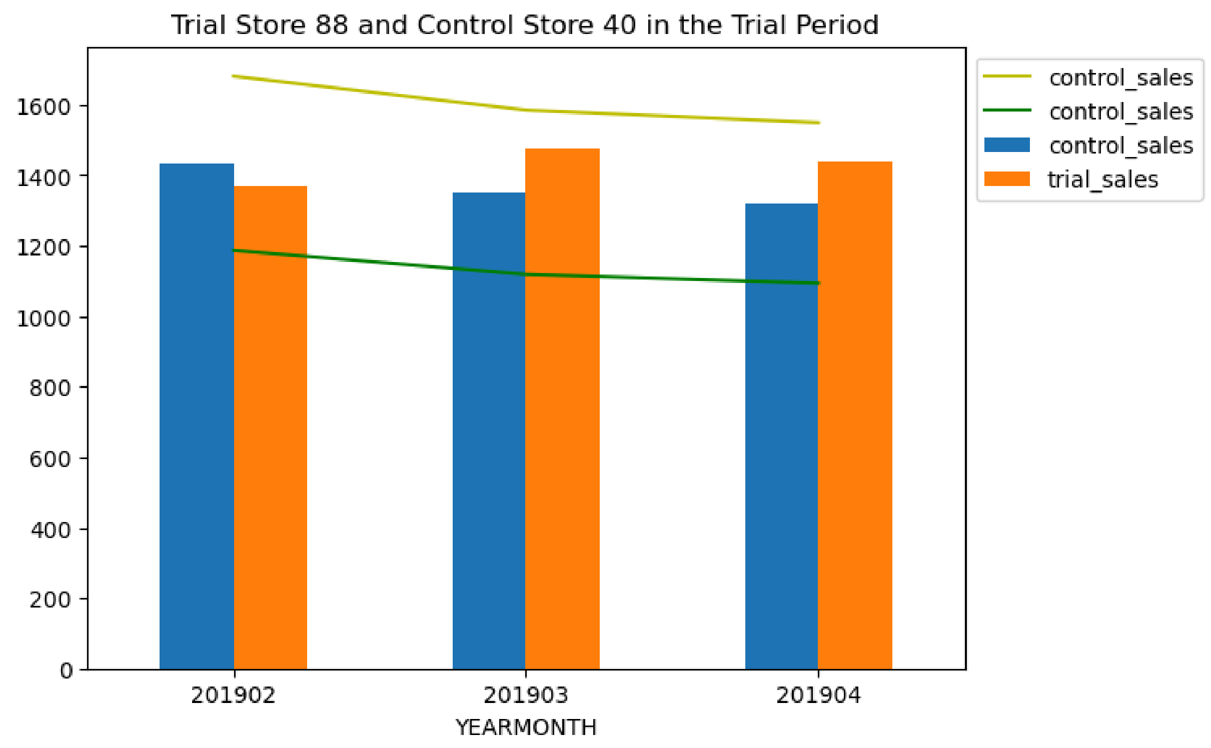
- **Sales of the control store**
- **Sales of the trial stores**
- **95th percentile value of sales of the control store**

By plotting these values, we can visually confirm the statistically significant increases in sales for trial stores 77 and 86.

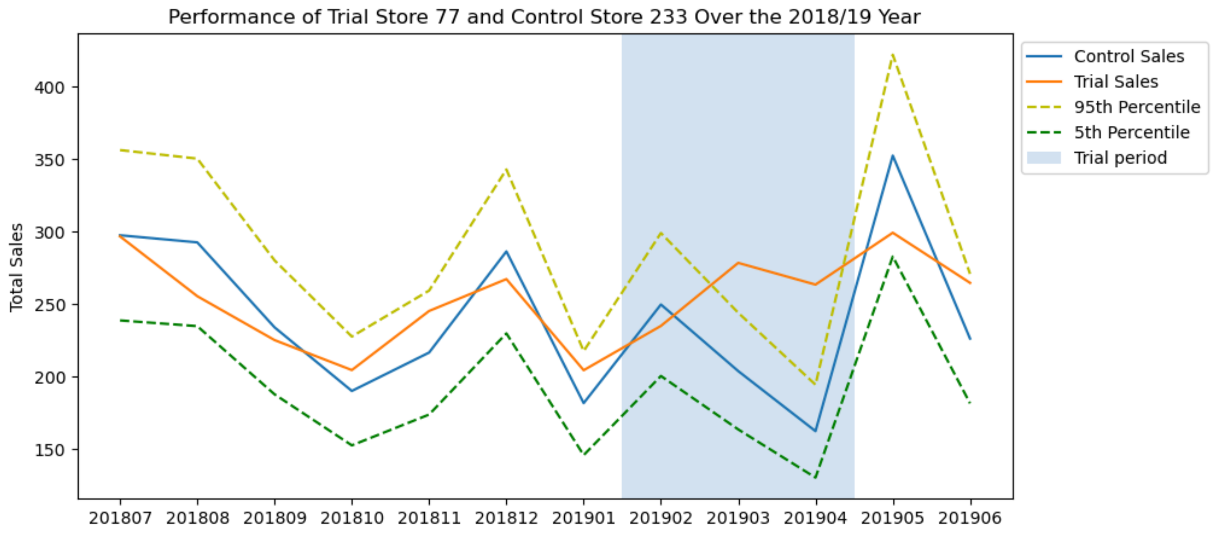
In [39]:
```
# First do bar graphs during the trial period
storepair = [[77, 233], [86, 155], [88, 40]]
for stores in storepair: # stores numbers are stored as [trial, control] in storepa
    trial = stores[0]
    control = stores[1]

    # Plot the bar chart of sales performance
    plot_control = percentdiff[(percentdiff['CONTROL_NBR'] == control) & (percentdi
                    [['YEARMONTH', 'CONTROL_NBR', 'scaled_sales_c']]
    plot_control = plot_control.rename(columns = {"CONTROL_NBR" : "STORE_NBR", "sca
    plot_trial = percentdiff[(percentdiff['TRIAL_NBR'] == trial) & (percentdiff.YEA
                    [['YEARMONTH', 'TRIAL_NBR', 'tot_sales_t']]
    plot_trial = plot_trial.rename(columns = {"TRIAL_NBR" : "STORE_NBR", "tot_sales
    toplot = plot_control[["YEARMONTH", "control_sales"]].merge(plot_trial[["YEARMO
    ax = toplot.plot(kind = 'bar',  figsize=(7, 5))


    # plot the thresholds as lines
    std = percentdiff[(percentdiff['CONTROL_NBR'] == control) & (percentdiff.YEARMO
    threshold95 = plot_control.reset_index()[['YEARMONTH', 'control_sales']]
    threshold95.control_sales = threshold95.control_sales*(1+std*2)
    threshold5 = plot_control.reset_index()[['YEARMONTH', 'control_sales']]
    threshold5.control_sales = threshold5.control_sales*(1-std*2)
    ax95 = threshold95.plot.line(x = 'YEARMONTH', y = 'control_sales',color='y', fi
    ax5 = threshold5.plot.line(x = 'YEARMONTH', y = 'control_sales', color='g', fig

    # Other plot features
    plt.legend(loc = "upper left",bbox_to_anchor=(1.0, 1.0))
    titlestr = 'Trial Store ' + str(trial) + ' and Control Store ' + str(control) +
    ax.set_title(titlestr)
    plt.show()
```

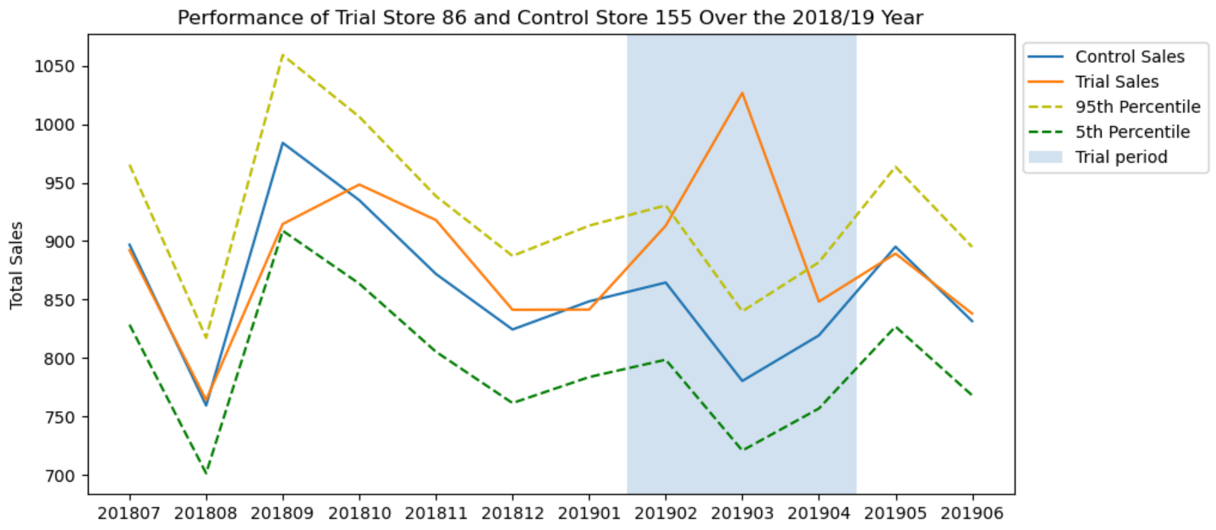


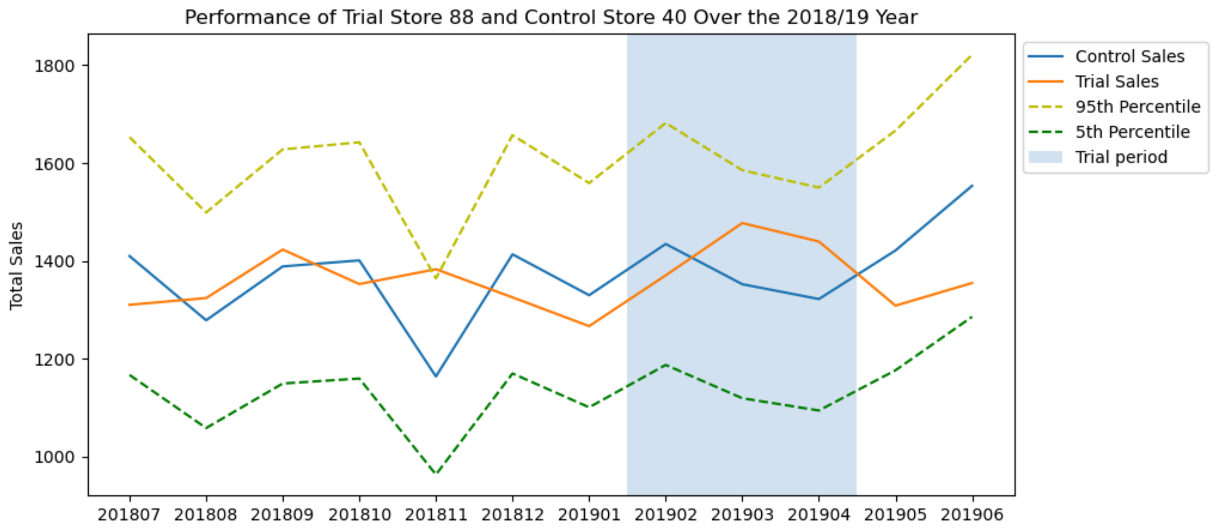




```
In [41]: # Then do line graphs during the whole year - for the report
         from matplotlib.patches import Rectangle
         storepair = [[77, 233], [86, 155], [88, 40]]
         for stores in storepair: # stores numbers are stored as [trial, control] in storepa
             trial = stores[0]
             control = stores[1]

             # Plot the line graph of sales performance
             plot_control = percentdiff[(percentdiff['CONTROL_NBR'] == control)][['YEARMONTH
             plot_control = plot_control.rename(columns = {"CONTROL_NBR" : "STORE_NBR", "sca
             plot_trial = percentdiff[(percentdiff['TRIAL_NBR'] == trial)][['YEARMONTH', 'TR
             plot_trial = plot_trial.rename(columns = {"TRIAL_NBR" : "STORE_NBR", "tot_sales

             ax = plot_control.plot.line(x = "YEARMONTH", y = 'control_sales', use_index=Fal
             ax_trial = plot_trial.plot.line(x = "YEARMONTH", y = 'trial_sales', use_index=F


             # plot the thresholds as lines
             std = percentdiff[(percentdiff['CONTROL_NBR'] == control) & (percentdiff.YEARMO
             threshold95 = plot_control.reset_index()[['YEARMONTH', 'control_sales']]
             threshold95.control_sales = threshold95.control_sales*(1+std*2)
             threshold5 = plot_control.reset_index()[['YEARMONTH', 'control_sales']]
             threshold5.control_sales = threshold5.control_sales*(1-std*2)
             ax95 = threshold95.plot.line(x = 'YEARMONTH', y = 'control_sales',color='y', li
             ax5 = threshold5.plot.line(x = 'YEARMONTH', y = 'control_sales', color='g',  li
             ax.add_patch(Rectangle((6.5, 0), 3, 2000, alpha = 0.2, label = 'Trial period'))

             # Other plot features
             ax.set_ylabel('Total Sales')
             plt.legend(loc = "upper left",bbox_to_anchor=(1.0, 1.0))
             titlestr = 'Performance of Trial Store ' + str(trial) + ' and Control Store ' +
             positions = (0,1,2,3,4,5,6,7,8,9, 10, 11)
             labels = ("201807", '201808', '201809', '201810', '201811', '201812', '201901',
             plt.xticks (positions, labels)
             ax.set_title(titlestr)
             plt.show()
```







The results show that the trial in store 77 is significantly 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.

For store 86, we can see that the trial in March is significantly different to the control store with the total sales performance outside of the 5% to 95% confidence interval. However, there is no significant difference in February's and April's performance.

The results for store 88 show no significant difference between the trial and control stores during this period.

Let's have a look at assessing this for number of customers as well.

In [43]:
```
# Calculate the scaling factor for the store pairs
scale_store77 = pretrial_metrics[pretrial_metrics.STORE_NBR == 77]['n_cust'].sum()/
scale_store86 = pretrial_metrics[pretrial_metrics.STORE_NBR == 86]['n_cust'].sum()/
scale_store88 = pretrial_metrics[pretrial_metrics.STORE_NBR == 88]['n_cust'].sum()/
```

In [44]:
```
# Extract the control store data from the df and scale according to the store
scaled_control233 = metrics_df[metrics_df.STORE_NBR.isin([233])][['STORE_NBR', "YEA
```

```
scaled_control233.n_cust *= scale_store77
scaled_control155 = metrics_df[metrics_df.STORE_NBR.isin([155])][['STORE_NBR', "YEA
scaled_control155.n_cust *= scale_store86
scaled_control40 = metrics_df[metrics_df.STORE_NBR.isin([40])][['STORE_NBR', "YEARM
scaled_control40.n_cust *= scale_store88

# Combine the scaled control stores to a single df
scaledncust_control = pd.concat([scaled_control233, scaled_control155, scaled_contr
scaledncust_control = scaledncust_control.rename(columns = {'n_cust':'scaled_n_cust
# Get the trial period of scaled control stores
scaledncust_control_trial = scaledncust_control[(scaledsales_control.YEARMONTH>=201

# Get the trial period of the trial stores
trialncust = metrics_df[metrics_df.STORE_NBR.isin([77,86,88])][['STORE_NBR', "YEARM
trialncust = trialncust.rename(columns = {'STORE_NBR': 'TRIAL_NBR'})
trialncust_trial = trialncust[(trialncust.YEARMONTH >= 201902) & (trialsales.YEARMO
```

In [45]:
```
# Calculate the percentage difference between the control and trial store pairs for
percentdiff = scaledncust_control.copy()
percentdiff[['TRIAL_NBR', 'n_cust_t']] = trialncust[['TRIAL_NBR', 'n_cust']]
percentdiff = percentdiff.rename(columns = {'scaled_n_cust' : 'scaled_n_cust_c'})
percentdiff['cust_percent_diff'] = (percentdiff.n_cust_t-percentdiff.scaled_n_cust_
                                    /(0.5*((percentdiff.scaled_n_cust_c+percentdiff
percentdiff.head()
```

Out[45]:

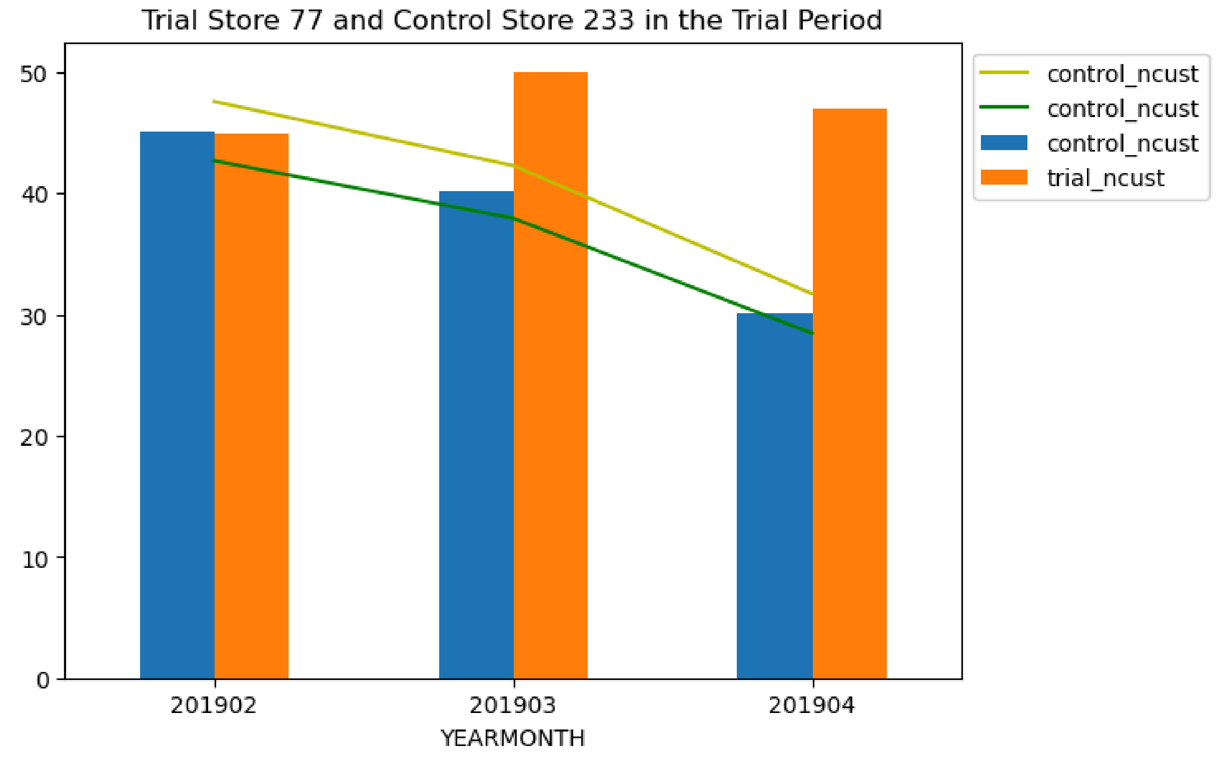
	CONTROL_NBR	YEARMONTH	scaled_n_cust_c	TRIAL_NBR	n_cust_t	cust_percent_diff
0	233	201807	51.171141	77	51	-0.003350
1	233	201808	48.161074	77	47	-0.024402
2	233	201809	42.140940	77	42	-0.003350
3	233	201810	35.117450	77	37	0.052208
4	233	201811	40.134228	77	41	0.021342

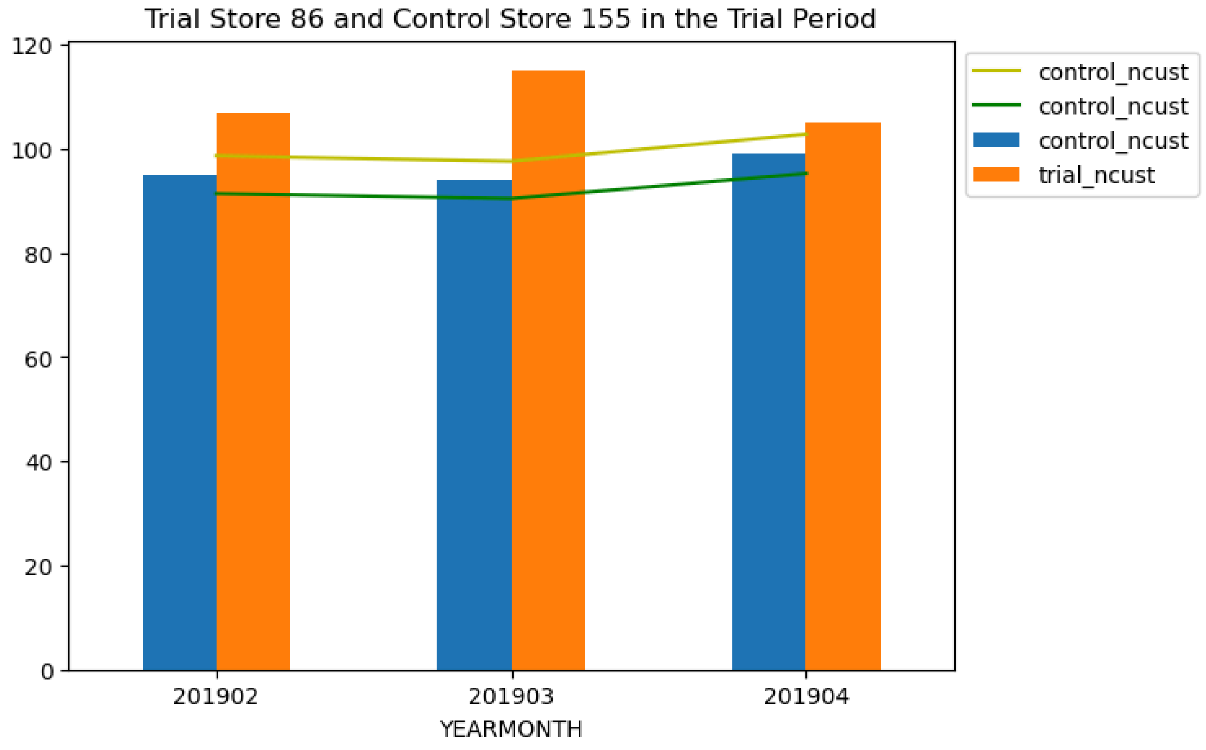
In [46]:
```
# As our null hypothesis is that the trial period is the same as the pre-trial peri
# let's take the standard deviation based on the scaled percentage difference in th
pretrial_percentdiff = percentdiff[percentdiff.YEARMONTH < 201902]
pretrial_percentdiff_std = pretrial_percentdiff.groupby(['TRIAL_NBR'])['cust_percen
dof = 6 # 7 months of data - 1

for stores in storepair: # stores numbers are stored as [trial, control] in storepa
    trialstore = stores[0]
    controlstore = stores[1]
    pretrial = percentdiff[(percentdiff.YEARMONTH < 201902) & (percentdiff.TRIAL_NB
    std = pretrial['cust_percent_diff'].agg('std')
    mean =  pretrial['cust_percent_diff'].agg('mean')
    trialperiod = percentdiff[(percentdiff.YEARMONTH >= 201902) & (percentdiff.YEAR
                              & (percentdiff.TRIAL_NBR == trialstore)]
    print("Trial store -", trialstore, "; control store -", controlstore)
    print("Month : t-statistic")
    for month in trialperiod.YEARMONTH.unique():
        xval = trialperiod[trialperiod.YEARMONTH == month]['cust_percent_diff'].ite
        tstat = ((xval - mean)/std)
        print(str(month), ' : ', tstat)
    print()

# Generate the t-statistic for the 95% percentile with 6 dof
print ('95th percentile value:', stats.t.ppf(1-0.05, 6))
```

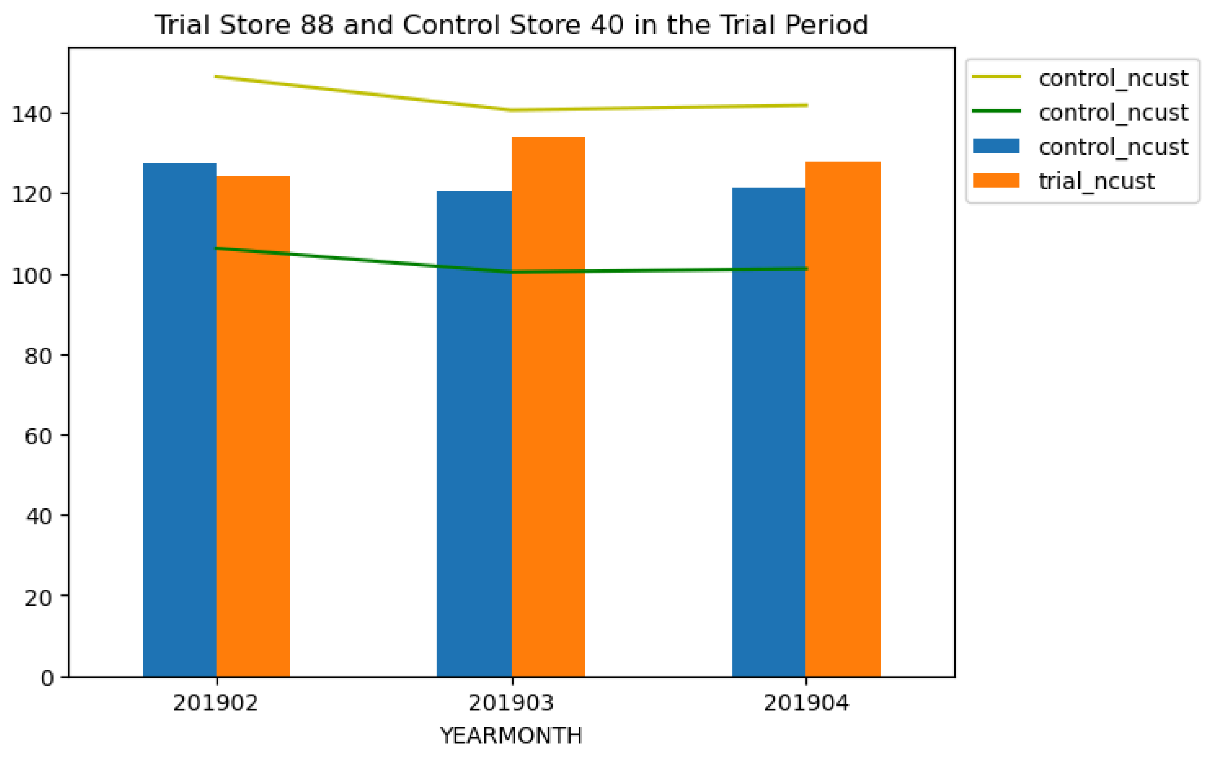
```
Trial store - 77 ; control store - 233
Month : t-statistic
201902  :  -0.19886295797440687
201903  :  8.009609025380932
201904  :  16.114474772873923

Trial store - 86 ; control store - 155
Month : t-statistic
201902  :  6.220524882227514
201903  :  10.52599074274189
201904  :  3.0763575852842706

Trial store - 88 ; control store - 40
Month : t-statistic
201902  :  -0.3592881735131531
201903  :  1.2575196020616801
201904  :  0.6092905590514273

95th percentile value: 1.9431802803927816
```

In [47]:

```
# First do bar charts to focus on the trial period
storepair = [[77, 233], [86, 155], [88, 40]]
for stores in storepair: # stores numbers are stored as [trial, control] in storepa
    trial = stores[0]
    control = stores[1]
    plot_control = percentdiff[(percentdiff['CONTROL_NBR'] == control) & (percentdi
                    [['YEARMONTH', 'CONTROL_NBR', 'scaled_n_cust_c']]
    plot_control = plot_control.rename(columns = {"CONTROL_NBR" : "STORE_NBR", "sca
    plot_trial = percentdiff[(percentdiff['TRIAL_NBR'] == trial) & (percentdiff.YEA
                    [['YEARMONTH', 'TRIAL_NBR', 'n_cust_t']]
    plot_trial = plot_trial.rename(columns = {"TRIAL_NBR" : "STORE_NBR", "n_cust_t"
    toplot = plot_control[["YEARMONTH", "control_ncust"]].merge(plot_trial[["YEARMO
    ax = toplot.plot(kind = 'bar',  figsize=(7, 5))


    # plot the thresholds as lines
    std = percentdiff[(percentdiff['CONTROL_NBR'] == control) & (percentdiff.YEARMO
    threshold95 = plot_control.reset_index()[['YEARMONTH', 'control_ncust']]
    threshold95.control_ncust = threshold95.control_ncust*(1+std*2)
    threshold5 = plot_control.reset_index()[['YEARMONTH', 'control_ncust']]
    threshold5.control_ncust = threshold5.control_ncust*(1-std*2)
    ax95 = threshold95.plot.line(x = 'YEARMONTH', y = 'control_ncust',color='y', fi
    ax5 = threshold5.plot.line(x = 'YEARMONTH', y = 'control_ncust', color='g', fig

    # Other plot features
    plt.legend(loc = "upper left",bbox_to_anchor=(1.0, 1.0))
    titlestr = 'Trial Store ' + str(trial) + ' and Control Store ' + str(control) +
    ax.set_title(titlestr)
    plt.show()
```

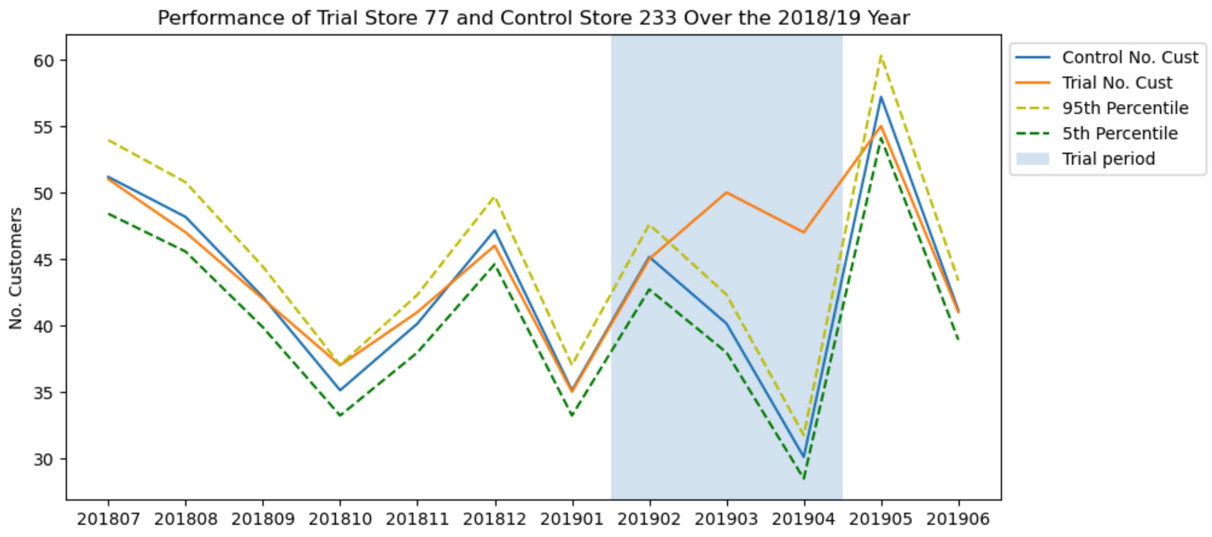

Trial Store 77 and Control Store 233 in the Trial Period
control_ncust
control_ncust
control_ncust
trial_ncust
50
40
30
20
10
0
201902
201903
201904
YEARMONTH

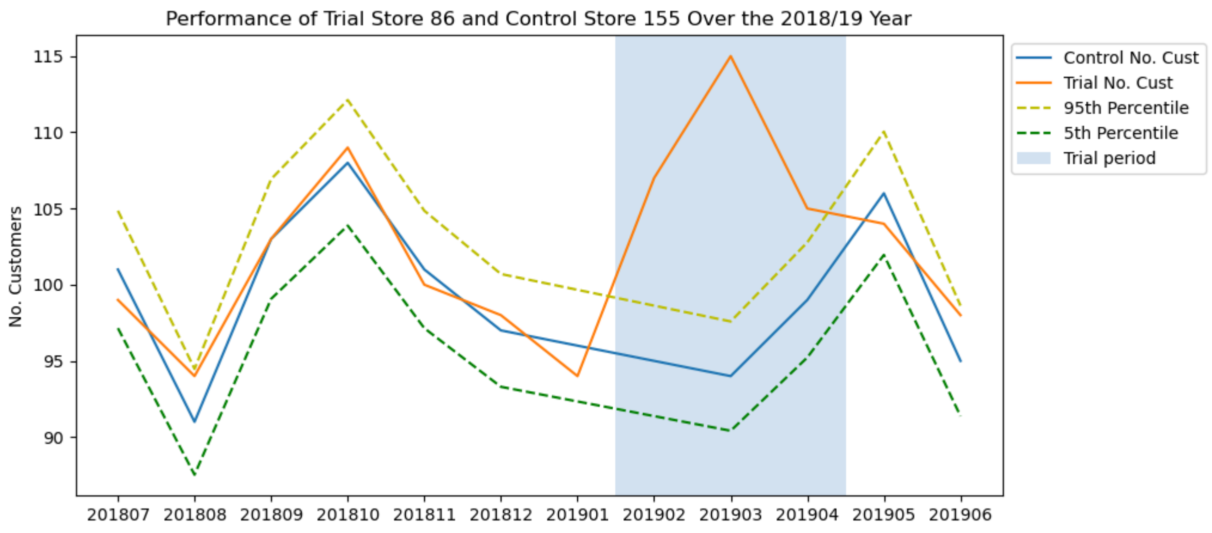

Trial Store 86 and Control Store 155 in the Trial Period
control_ncust
control_ncust
control_ncust
trial_ncust
120
100
80
60
40
20
0
201902
201903
201904
YEARMONTH

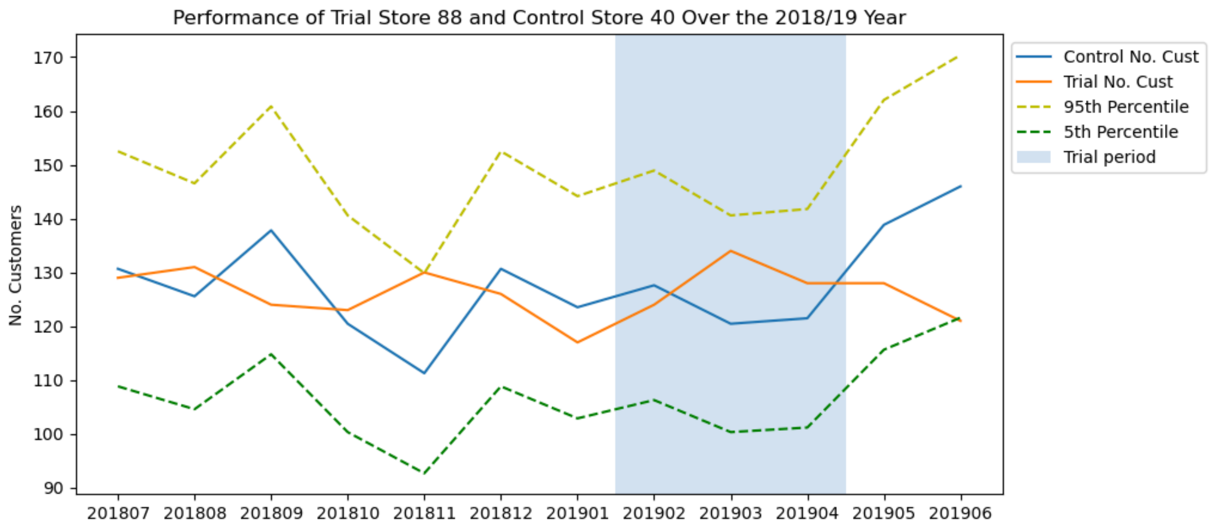


```
In [48]: storepair = [[77, 233], [86, 155], [88, 40]]
         for stores in storepair: # stores numbers are stored as [trial, control] in storepa
             trial = stores[0]
             control = stores[1]
             plot_control = percentdiff[(percentdiff['CONTROL_NBR'] == control)]\
                             [['YEARMONTH', 'CONTROL_NBR', 'scaled_n_cust_c']]
             plot_control = plot_control.rename(columns = {"CONTROL_NBR" : "STORE_NBR", "sca
             plot_trial = percentdiff[(percentdiff['TRIAL_NBR'] == trial)]\
                             [['YEARMONTH', 'TRIAL_NBR', 'n_cust_t']]
             plot_trial = plot_trial.rename(columns = {"TRIAL_NBR" : "STORE_NBR", "n_cust_t"

             ax = plot_control.plot.line(x = "YEARMONTH", y = 'control_ncust', use_index=Fal
             ax_trial = plot_trial.plot.line(x = "YEARMONTH", y = 'trial_ncust', use_index=F

             # plot the thresholds as lines
             std = percentdiff[(percentdiff['CONTROL_NBR'] == control) & (percentdiff.YEARMO
             threshold95 = plot_control.reset_index()[['YEARMONTH', 'control_ncust']]
             threshold95.control_ncust = threshold95.control_ncust*(1+std*2)
             threshold5 = plot_control.reset_index()[['YEARMONTH', 'control_ncust']]
             threshold5.control_ncust = threshold5.control_ncust*(1-std*2)
             ax95 = threshold95.plot.line(x = 'YEARMONTH', y = 'control_ncust',color='y', li
             ax5 = threshold5.plot.line(x = 'YEARMONTH', y = 'control_ncust', color='g',  li
             ax.add_patch(Rectangle((6.5, 0), 3, 2000, alpha = 0.2, label = 'Trial period'))

             # Other plot features
             ax.set_ylabel('No. Customers')
             plt.legend(loc = "upper left",bbox_to_anchor=(1.0, 1.0))
             titlestr = 'Performance of Trial Store ' + str(trial) + ' and Control Store ' +
             positions = (0,1,2,3,4,5,6,7,8,9, 10, 11)
             labels = ("201807", '201808', '201809', '201810', '201811', '201812', '201901',
             plt.xticks (positions, labels)
```

```
    ax.set_title(titlestr)
    plt.show()
```







It looks like the number of customers is significantly higher in all of the three months for store 77 and 86. 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, the statistical significance in the total sales were not as large, compared to store 77. We should check with the Category Manager if there were special deals in the trial store that were may have resulted in lower prices, impacting the results. Likewise to when considering the total sales, there appears to be no significant different in the number of customers between the control and trial stores for store 88 over the trial period.

Conclusions

In this task, we found that the results for trial stores 77 and 86 showed a statistically significant difference in at least two stores of the three months of the trial period. However, this was not the case for store 88. We can check to see if the trial was implemented differently in store 88 but even so, we have been able to see that the trial has resulted in a significant increase in sales.

```
In [ ]:
```