Experimentation and uplift testing

Julia has asked us to evaluate the performance of a store trial which was performed in stores 77, 86 and 88.

We have chosen to complete this task in R, however you will also find Python to be a useful tool in this piece of analytics. We have also provided an R solution template if you want some assistance in getting through this Task.

To get started use the QVI_data dataset below or your output from task 1 and consider the monthly sales experience of each store.

This can be broken down by:

1. total sales revenue
2. total number of customers
3. average number of transactions per customer

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. Consider using Pearson correlations or a metric such as a magnitude distance e.g. 1- (Observed distance – minimum distance)/(Maximum distance – minimum distance) as a measure.

Once you have selected your control stores, compare each trial and control pair during the trial period. You 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 etc.

Main areas of focus are :

1. Select control stores – Explore data, define metrics, visualize graphs
2. Assessment of the trial – insights/trends by comparing trial stores with control stores
3. Collate findings – summarize and provide recommendations

Import the necessary packages

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import datetime
import matplotlib.pyplot as plt
import re
```

In [2]:
```
# Import the QVI_transaction_data dataset into Jupyter Notebooks

file_location = 'C:\\Users\\DELL\\Desktop\\QVI DATA\\'
customer_transaction_data = pd.read_excel(file_location + 'customer_transaction_da
```

In [3]:
```
# View the first 5 rows of our dataframe
```

```
customer_transaction_data.head()
```

Out[3]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SAL
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt	2	6
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese	3	6
2	2018-11-10	1	1307	346	96	WW Original Stacked Chips	2	3
3	2019-03-09	1	1307	347	54	CCs Original	1	2
4	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken	2	2

In [4]:
```
# View the first 5 rows of our dataframe

customer_transaction_data.tail()
```

Out[4]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TO
246735	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream	2	
246736	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime	1	
246737	2018-11-06	272	272379	270187	51	Doritos Mexicana	2	
246738	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno	2	
246739	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime	2	

In [ ]:
```

```

Check for nulls

In [5]:
```
customer_transaction_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246740 entries, 0 to 246739
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   DATE              246740 non-null  datetime64[ns]
 1   STORE_NBR         246740 non-null  int64
 2   LYLTY_CARD_NBR    246740 non-null  int64
 3   TXN_ID            246740 non-null  int64
 4   PROD_NBR          246740 non-null  int64
 5   PROD_NAME         246740 non-null  object
 6   PROD_QTY          246740 non-null  int64
 7   TOT_SALES         246740 non-null  float64
 8   PACK_SIZE_GRAMS   246740 non-null  int64
 9   BRAND             246740 non-null  object
 10  LIFESTAGE         246740 non-null  object
 11  PREMIUM_CUSTOMER  246740 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(6), object(4)
memory usage: 22.6+ MB
```

In [ ]:

The client has selected store numbers 77, 86 and 88 as trial stores and want 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

Let's first create the metrics of interest and filter to stores that are present throughout the pre-trial period.

Let's add a new month ID, YEARMONTH,column in the data with the format yyyymm

In [6]:
```
# create new column YEARMONTH in format yyyymm
customer_transaction_data["DATE"] = pd.to_datetime(customer_transaction_data["DATE
customer_transaction_data["YEARMONTH"] = customer_transaction_data["DATE"].dt.strf

# View the created MONTH column
customer_transaction_data['YEARMONTH']
```

Out[6]:
```
0         201810
1         201905
2         201811
3         201903
4         201905
           ...
246735    201903
246736    201808
246737    201811
246738    201812
246739    201809
Name: YEARMONTH, Length: 246740, dtype: int64
```

In [ ]:

Create a new data frame measureOverTime to calculate total sales, number of customers, transactions per customer, chips per customer and the average price per unit fore each store and month

In [7]:
```
# Group the data by STORE_NBR and YEARMONTH, and calculate the required metrics

def measureOverTime():
    store_YEARMONTH_group = customer_transaction_data.groupby(["STORE_NBR", "YEARM
    total_sales = store_YEARMONTH_group["TOT_SALES"].sum()
    num_cust = store_YEARMONTH_group["LYLTY_CARD_NBR"].nunique()
    trans_per_cust = store_YEARMONTH_group.size() / num_cust
    avg_chips_per_cust = store_YEARMONTH_group["PROD_QTY"].sum() / num_cust
    avg_chips_price = total_sales / store_YEARMONTH_group["PROD_QTY"].sum()
    aggregates = [total_sales, num_cust, trans_per_cust, avg_chips_per_cust, avg_cl
    metrics = pd.concat(aggregates, axis = 1)
    metrics.columns = ["TOT_SALES", "nCustomers", "nTxnPerCust", "nChipsPerTxn", "a
    return metrics
```

In [8]:
```
store_monthly_metrics = measureOverTime().reset_index()
store_monthly_metrics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3165 entries, 0 to 3164
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   STORE_NBR        3165 non-null   int64  
 1   YEARMONTH        3165 non-null   int64  
 2   TOT_SALES        3165 non-null   float64
 3   nCustomers       3165 non-null   int64  
 4   nTxnPerCust      3165 non-null   float64
 5   nChipsPerTxn     3165 non-null   float64
 6   avgPricePerUnit  3165 non-null   float64
dtypes: float64(4), int64(3)
memory usage: 173.2 KB
```

In [ ]:
```
```

Filter to the pre-trial period and stores with full observation periods (12 months) - control stores

In [9]:
```
store_full_obs = store_monthly_metrics["STORE_NBR"].value_counts()
full_observ_index = store_full_obs[store_full_obs == 12].index
full_observ = store_monthly_metrics[store_monthly_metrics["STORE_NBR"].isin(full_ob
preTrialMeasures = full_observ[full_observ["YEARMONTH"] < 201902]

preTrialMeasures.head(10)
```

Out[9]:

	STORE_NBR	YEARMONTH	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerU
0	1	201807	188.9	47	1.042553	1.234043	3.256
1	1	201808	168.4	41	1.000000	1.268293	3.238
2	1	201809	268.1	57	1.035088	1.245614	3.776
3	1	201810	175.4	39	1.025641	1.307692	3.439
4	1	201811	184.8	44	1.022727	1.250000	3.360
5	1	201812	160.6	37	1.081081	1.297297	3.345
6	1	201901	149.7	35	1.000000	1.171429	3.651
12	2	201807	140.5	36	1.055556	1.194444	3.267
13	2	201808	180.9	35	1.114286	1.428571	3.618
14	2	201809	133.9	32	1.031250	1.125000	3.719

In [10]:
```
preTrialMeasures.shape
```

Out[10]:
```
(1813, 7)
```

In [ ]:
```

```

Create a function to calculate correlation for a measure, looping through each control store.

In [11]:
```
def calculate_correlation(measure, storeComparison, inputTable=preTrialMeasures):
    control_store_nbrs = inputTable[~inputTable["STORE_NBR"].isin([77, 86, 88])]["
    corrs = pd.DataFrame(columns = ["YEARMONTH", "Trial_Str", "Ctrl_Str", "Corr_Sc
    trial_store = inputTable[inputTable["STORE_NBR"] == storeComparison][measure].
    for control in control_store_nbrs:
        concat_df = pd.DataFrame(columns = ["YEARMONTH", "Trial_Str", "Ctrl_Str",
        control_store = inputTable[inputTable["STORE_NBR"] == control][measure].re
        concat_df["Corr_Score"] = trial_store.corrwith(control_store, axis=1)
        concat_df["Trial_Str"] = storeComparison
        concat_df["Ctrl_Str"] = control
        concat_df["YEARMONTH"] = list(inputTable[inputTable["STORE_NBR"] == storeC
        corrs = pd.concat([corrs, concat_df])
    return corrs
```

In [12]:
```
corr_table = pd.DataFrame()
for trial_num in [77, 86, 88]:
    corr_table = pd.concat([corr_table, calculate_correlation(["TOT_SALES", "nCust

corr_table.head(10)
```

Out[12]:

	YEARMONTH	Trial_Str	Ctrl_Str	Corr_Score
0	201807	77	1	0.034928
1	201808	77	1	0.017017
2	201809	77	1	-0.007445
3	201810	77	1	-0.028881
4	201811	77	1	0.007549
5	201812	77	1	0.051580
6	201901	77	1	-0.017382
0	201807	77	2	0.116765
1	201808	77	2	0.094303
2	201809	77	2	0.080021

In [ ]:

Create a function to calculate a standardised magnitude distance for a measure looping through each control store

In [13]:
```
def calculate_magnitude_dist(measure, storeComparison, inputTable=preTrialMeasures
    control_store_nbrs = inputTable[~inputTable["STORE_NBR"].isin([77, 86, 88])]["
    dists = pd.DataFrame()
    trial_store = inputTable[inputTable["STORE_NBR"] == storeComparison][measure]
    for control in control_store_nbrs:
        concat_df = abs(inputTable[inputTable["STORE_NBR"] == storeComparison].res
        concat_df["YEARMONTH"] = list(inputTable[inputTable["STORE_NBR"] == storeC
        concat_df["Trial_Str"] = storeComparison
        concat_df["Ctrl_Str"] = control
        dists = pd.concat([dists, concat_df])
    for col in measure:
        dists[col] = 1 - ((dists[col] - dists[col].min()) / (dists[col].max() - di
    dists["magnitude"] = dists[measure].mean(axis=1)
    return dists
```

In [14]:
```
dist_table = pd.DataFrame()
for trial_num in [77, 86, 88]:
    dist_table = pd.concat([dist_table, calculate_magnitude_dist(["TOT_SALES", "nC

dist_table.head(8)
dist_table
```

Out[14]:

	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	YEARMONTH	Trial_Str
0	0.939141	1.000000	0.906165	0.716452	0.853885	201807	77
1	0.939447	0.950495	1.000000	0.807168	0.841984	201808	77
2	0.960729	0.831683	0.967117	0.716360	0.686332	201809	77
3	0.985532	0.970297	0.995288	0.943513	0.714699	201810	77
4	0.969303	0.950495	0.880491	0.743772	0.866003	201811	77
...	...	...	...	...	...	...	...
2	0.209407	0.268293	0.613056	0.797921	0.923369	201809	88
3	0.358741	0.357724	0.773519	0.861789	0.969082	201810	88
4	0.292148	0.317073	0.665863	0.792787	0.964449	201811	88
5	0.370444	0.373984	0.571429	0.722481	0.963101	201812	88
6	0.390370	0.422764	0.548842	0.728854	0.978972	201901	88

5376 rows × 9 columns

In [ ]:

Create a combined score composed of correlation and magnitude, by first merging the correlations table with the magnitude table.

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 by using correlation and magnitude distance.

In [15]:
```
def combine_corr_dist(measure, storeComparison, inputTable=preTrialMeasures):
    corrs = calculate_correlation(measure, storeComparison, inputTable)
    dists = calculate_magnitude_dist(measure, storeComparison, inputTable)
    dists = dists.drop(measure, axis=1)
    combine = pd.merge(corrs, dists, on=["YEARMONTH", "Trial_Str", "Ctrl_Str"])
    return combine
```

In [16]:
```
compare_metrics_table1 = pd.DataFrame()
for trial_num in [77, 86, 88]:
    compare_metrics_table1 = pd.concat([compare_metrics_table1, combine_corr_dist(
```

In [17]:
```
corr_weight = 0.5
dist_weight = 1 - corr_weight
```

In [ ]:

Let's determine the top five highest composite score for each trial based on total sales

In [18]:
```
grouped_comparison_table1 = compare_metrics_table1.groupby(["Trial_Str", "Ctrl_Str
grouped_comparison_table1["CompScore"] = (corr_weight * grouped_comparison_table1[
for trial_num in compare_metrics_table1["Trial_Str"].unique():
    print(grouped_comparison_table1[grouped_comparison_table1["Trial_Str"] == tria
```

```
     Trial_Str  Ctrl_Str  Corr_Score  magnitude  CompScore
217         77       233         1.0   0.986702   0.993351
176         77       188         1.0   0.980567   0.990283
120         77       131         1.0   0.977341   0.988670
238         77       255         1.0   0.976794   0.988397
192         77       205         1.0   0.976706   0.988353

     Trial_Str  Ctrl_Str  Corr_Score  magnitude  CompScore
400         86       155         1.0   0.964165   0.982082
355         86       109         1.0   0.955015   0.977507
383         86       138         1.0   0.950843   0.975421
362         86       116         1.0   0.950053   0.975027
449         86       207         1.0   0.949940   0.974970

     Trial_Str  Ctrl_Str  Corr_Score  magnitude  CompScore
549         88        40         1.0   0.933101   0.966550
515         88         4         1.0   0.911153   0.955577
536         88        26         1.0   0.907058   0.953529
566         88        58         1.0   0.893688   0.946844
580         88        72         1.0   0.891545   0.945772
```

Similarities based on total sales:

- Trial store 77: Stores 233, 188, 131
- Trial store 86: Stores 155, 109, 138
- Trial store 88: Stores 40, 4, 26

In [19]:
```
compare_metrics_table2 = pd.DataFrame()
for trial_num in [77, 86, 88]:
    compare_metrics_table2 = pd.concat([compare_metrics_table2, combine_corr_dist(
```

In [ ]:
```

```

Let's determine the top five highest composite score for each trial based on no. of customers

In [20]:
```
grouped_comparison_table2 = compare_metrics_table2.groupby(["Trial_Str", "Ctrl_Str
grouped_comparison_table2["CompScore"] = (corr_weight * grouped_comparison_table2[
for trial_num in compare_metrics_table2["Trial_Str"].unique():
    print(grouped_comparison_table2[grouped_comparison_table2["Trial_Str"] == tria
```

```
     Trial_Str  Ctrl_Str  Corr_Score  magnitude  CompScore
217         77       233         1.0   0.991513   0.995757
38          77        41         1.0   0.961810   0.980905
15          77        17         1.0   0.960396   0.980198
42          77        46         1.0   0.960396   0.980198
134         77       145         1.0   0.957567   0.978784

     Trial_Str  Ctrl_Str  Corr_Score  magnitude  CompScore
400         86       155         1.0   0.975035   0.987517
465         86       225         1.0   0.968100   0.984050
360         86       114         1.0   0.959778   0.979889
355         86       109         1.0   0.951456   0.975728
428         86       184         1.0   0.951456   0.975728

     Trial_Str  Ctrl_Str  Corr_Score  magnitude  CompScore
733         88       237         1.0   0.979094   0.989547
549         88        40         1.0   0.939605   0.969803
698         88       199         1.0   0.938444   0.969222
666         88       165         1.0   0.931475   0.965738
700         88       201         1.0   0.924506   0.962253
```

Similarities based on no. of customers:

- Trial store 77: Stores 233, 41, 17
- Trial store 86: Stores 155, 225, 114
- Trial store 88: Stores 237, 40, 199

In [ ]:

Let's determine the top composite score for each trial based on highest average total sales and no. of customers

In [21]:
```
for trial_num in compare_metrics_table2["Trial_Str"].unique():
    a = grouped_comparison_table1[grouped_comparison_table1["Trial_Str"] == trial_
    b = grouped_comparison_table2[grouped_comparison_table2["Trial_Str"] == trial_
    print((pd.concat([a,b], axis=1).sum(axis=1)/2).sort_values(ascending=False).hea
```

```
Trial_Str  Ctrl_Str
77         233         0.994554
           46          0.983852
           188         0.981705
dtype: float64

Trial_Str  Ctrl_Str
86         155         0.984800
           109         0.976618
           225         0.975346
dtype: float64

Trial_Str  Ctrl_Str
88         40          0.968176
           26          0.957020
           58          0.953097
dtype: float64
```

Final similarities based on highest average of both features combined:

- Trial store 77: Store 233

- Trial store 86: Store 155
- Trial store 88: Store 40

In [ ]:

Similarities based on total sales:

- Trial store 77: Stores 233, 188, 131
- Trial store 86: Stores 155, 109, 138
- Trial store 88: Stores 40, 4, 26

Similarities based on no. of customers:

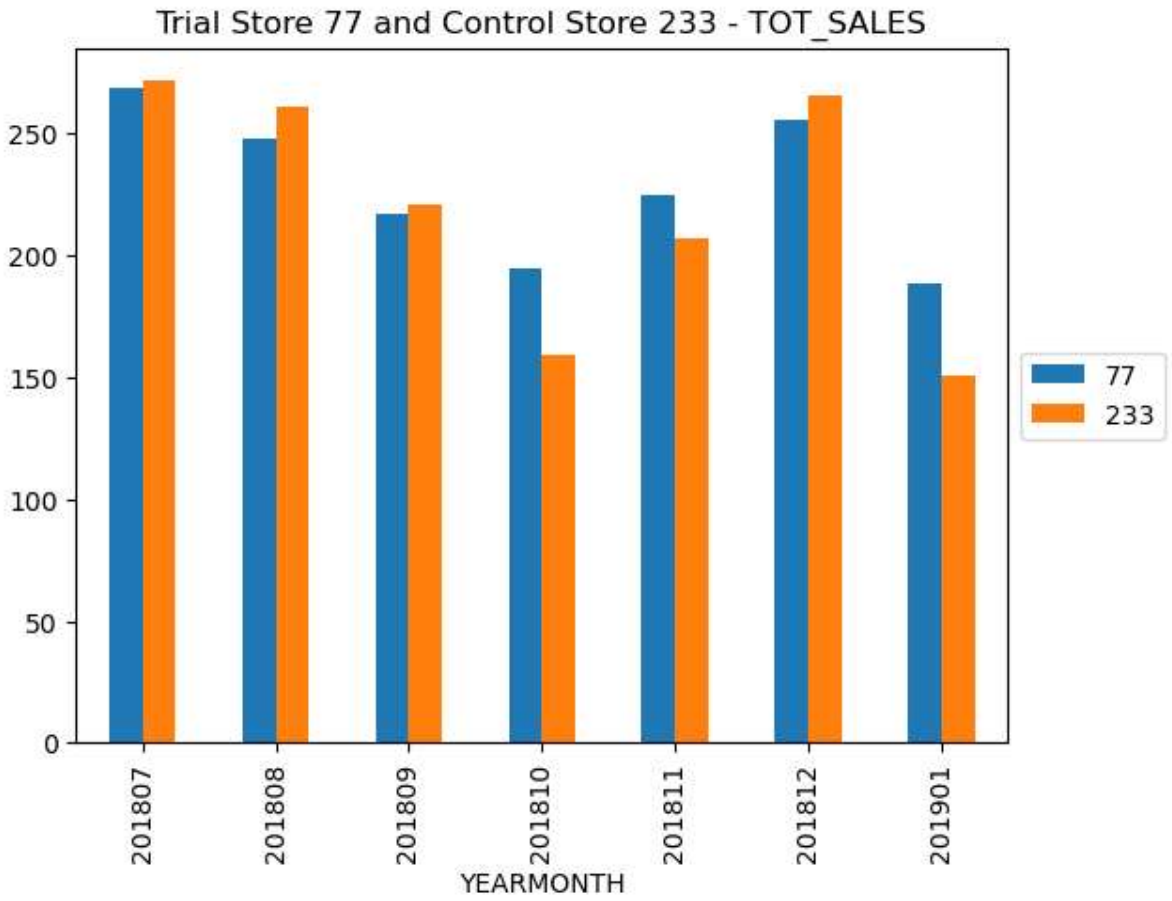
- Trial store 77: Stores 233, 41, 17
- Trial store 86: Stores 155, 225, 114
- Trial store 88: Stores 237, 40, 199

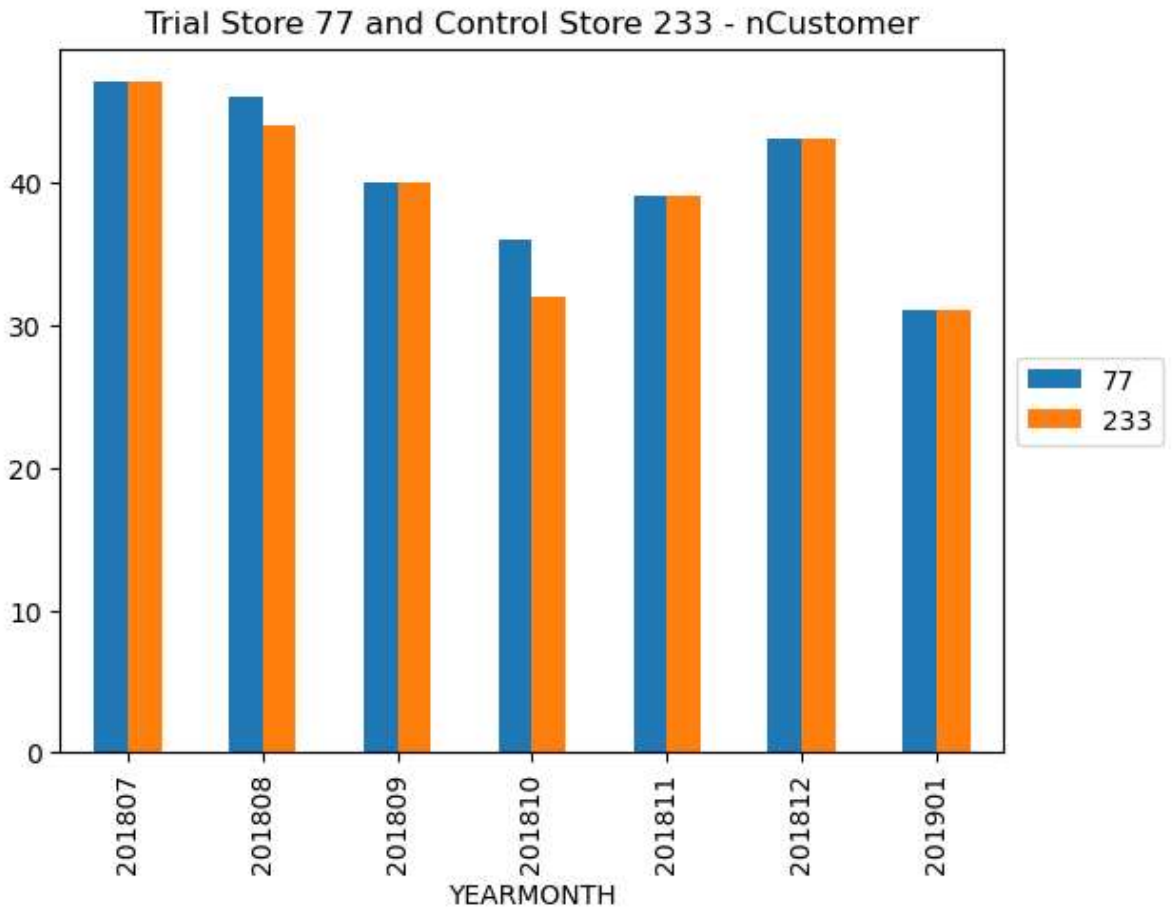
Final similarities based on highest average of both features combined:

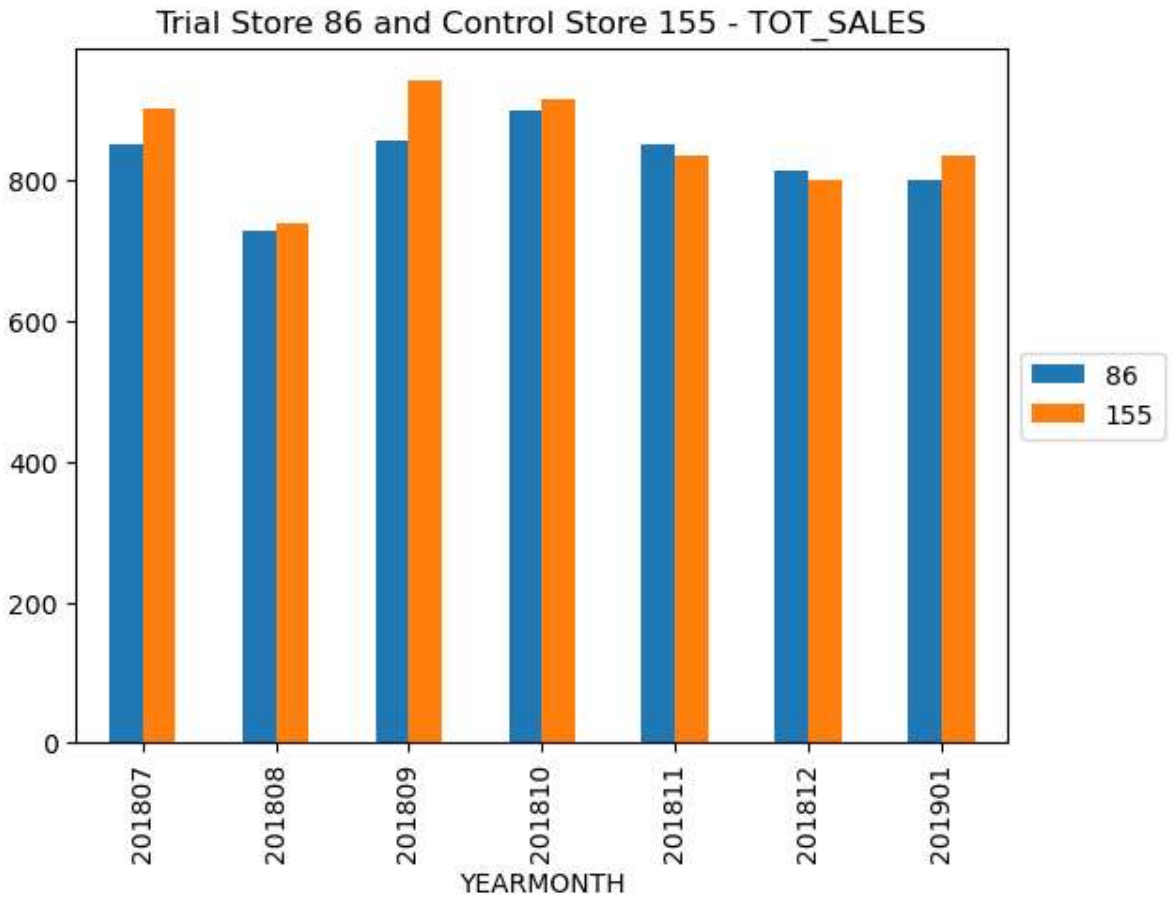
- Trial store 77: Store 233
- Trial store 86: Store 155
- Trial store 88: Store 40

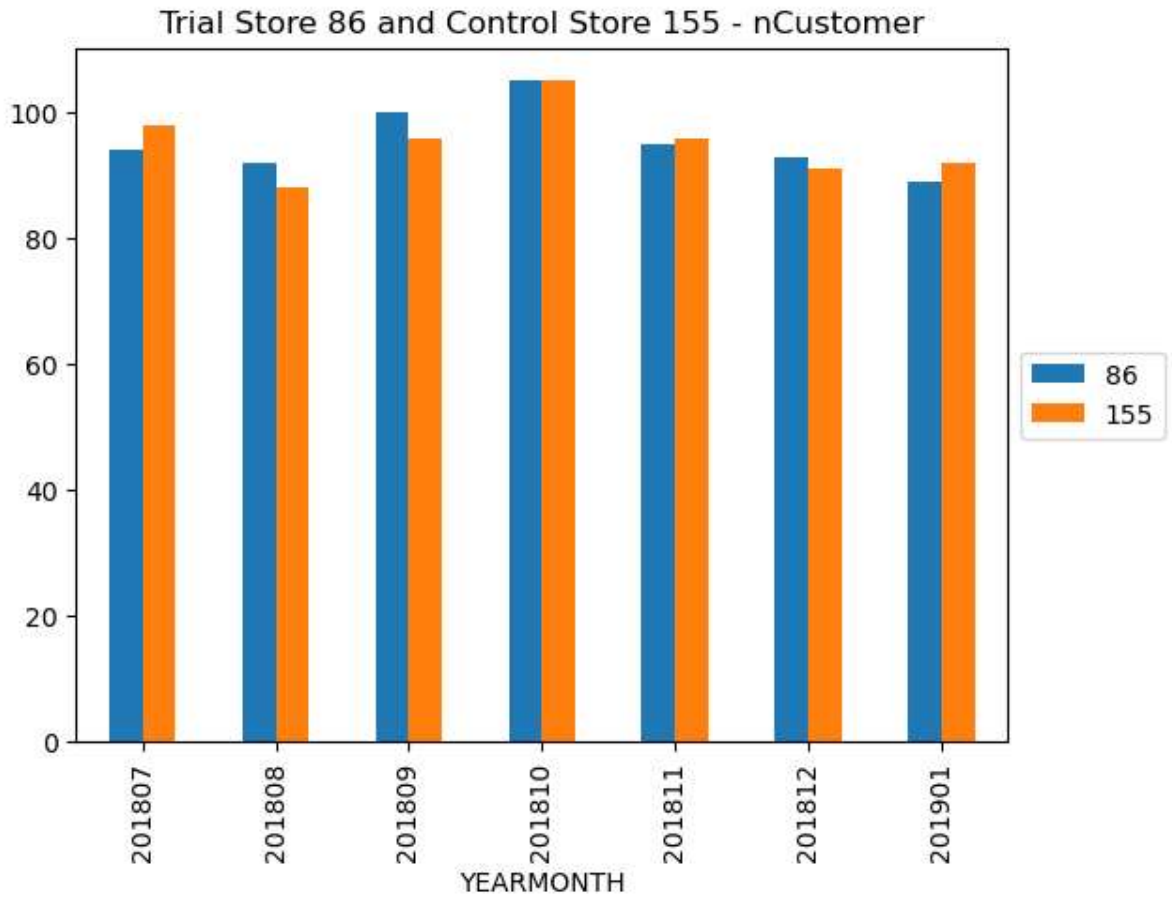
In [ ]:

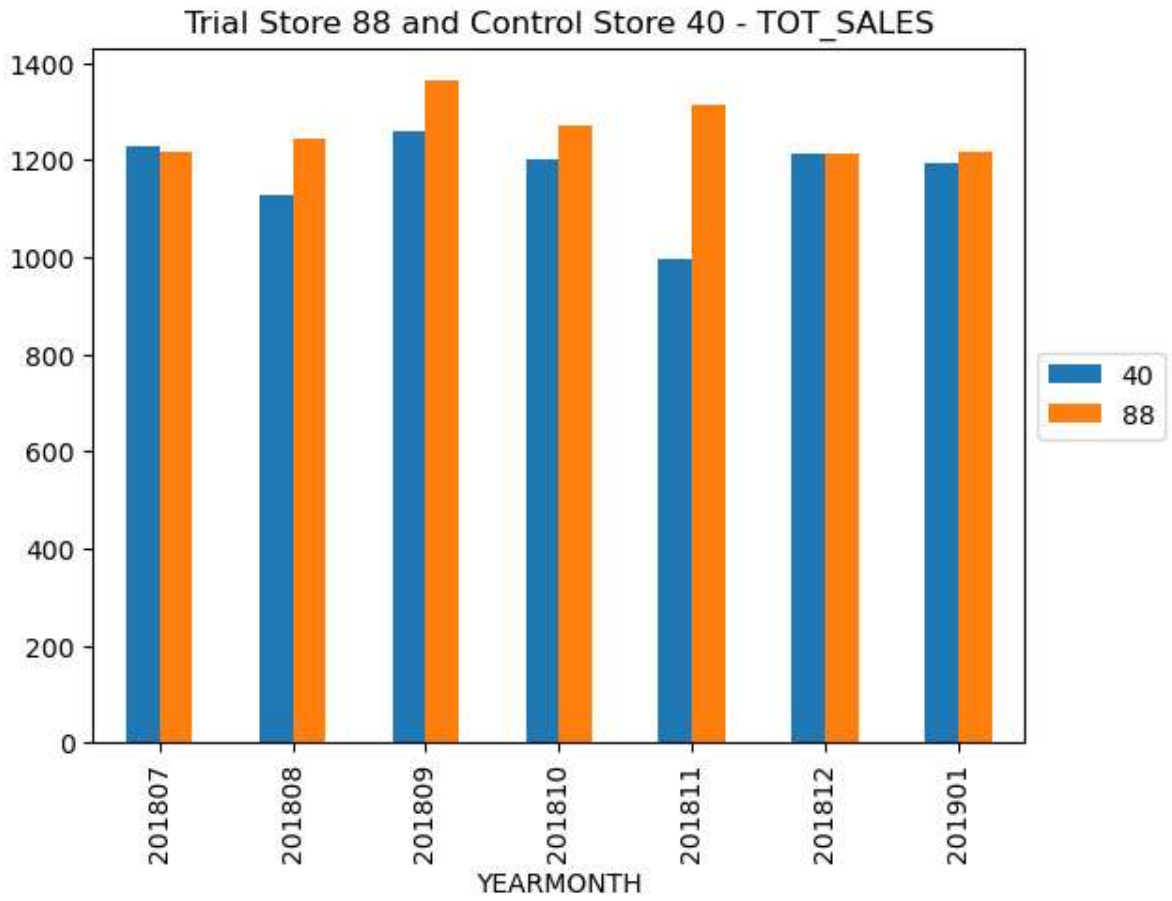
In [22]:
```
trial_control_dic = {77:233, 86:155, 88:40}
for key, val in trial_control_dic.items():
    preTrialMeasures[preTrialMeasures["STORE_NBR"].isin([key, val])].groupby(
        ["YEARMONTH", "STORE_NBR"]).sum()["TOT_SALES"].unstack().plot.bar()
    plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
    plt.title("Trial Store "+str(key)+" and Control Store "+str(val)+" - TOT_SALES
    plt.show()
    preTrialMeasures[preTrialMeasures["STORE_NBR"].isin([key, val])].groupby(
    ["YEARMONTH", "STORE_NBR"]).sum()["nCustomers"].unstack().plot.bar()
    plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
    plt.title("Trial Store "+str(key)+" and Control Store "+str(val)+" - nCustomer
    plt.show()
```


Trial Store 77 and Control Store 233 - TOT_SALES
250
200
150
100
50
0
201807
201808
201809
201810
201811
201812
201901
YEARMONTH
77
233


Trial Store 77 and Control Store 233 - nCustomer
40
30
20
10
0
201807
201808
201809
201810
201811
201812
201901
YEARMONTH
77
233


Trial Store 86 and Control Store 155 - TOT_SALES
800
600
400
200
0
201807
201808
201809
201810
201811
201812
201901
YEARMONTH
86
155


Trial Store 86 and Control Store 155 - nCustomer
100
80
60
40
20
0
201807
201808
201809
201810
201811
201812
201901
YEARMONTH
86
155



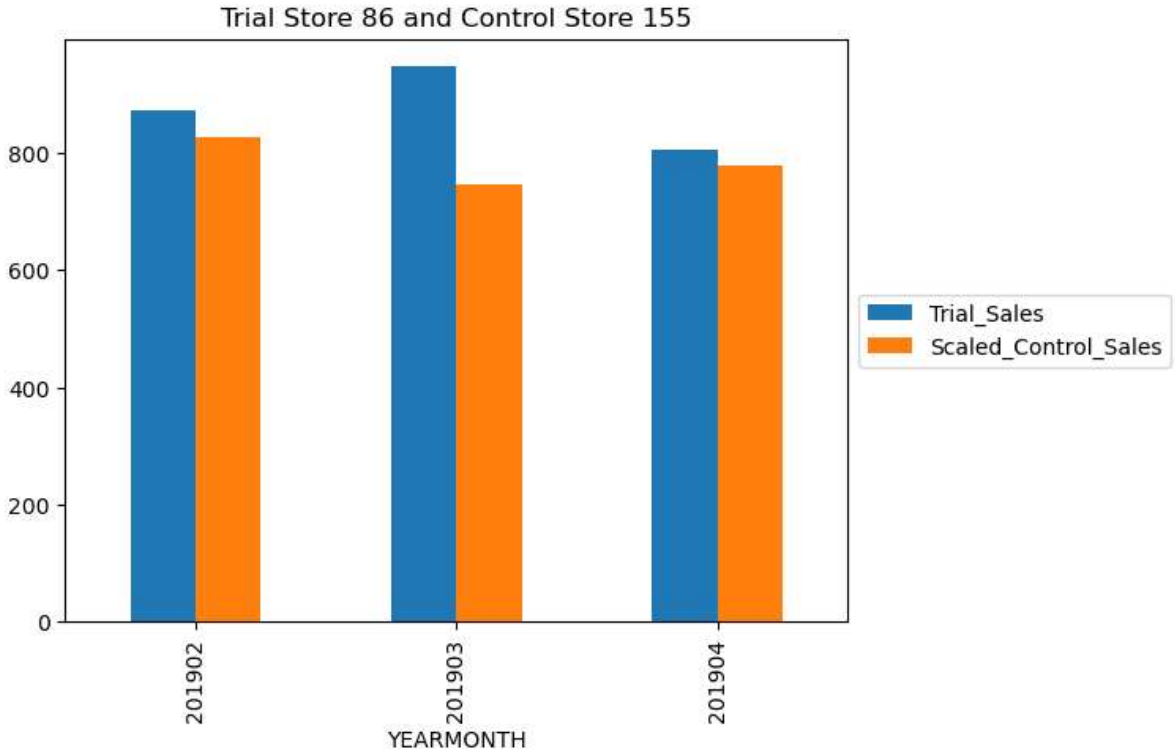
In [ ]:

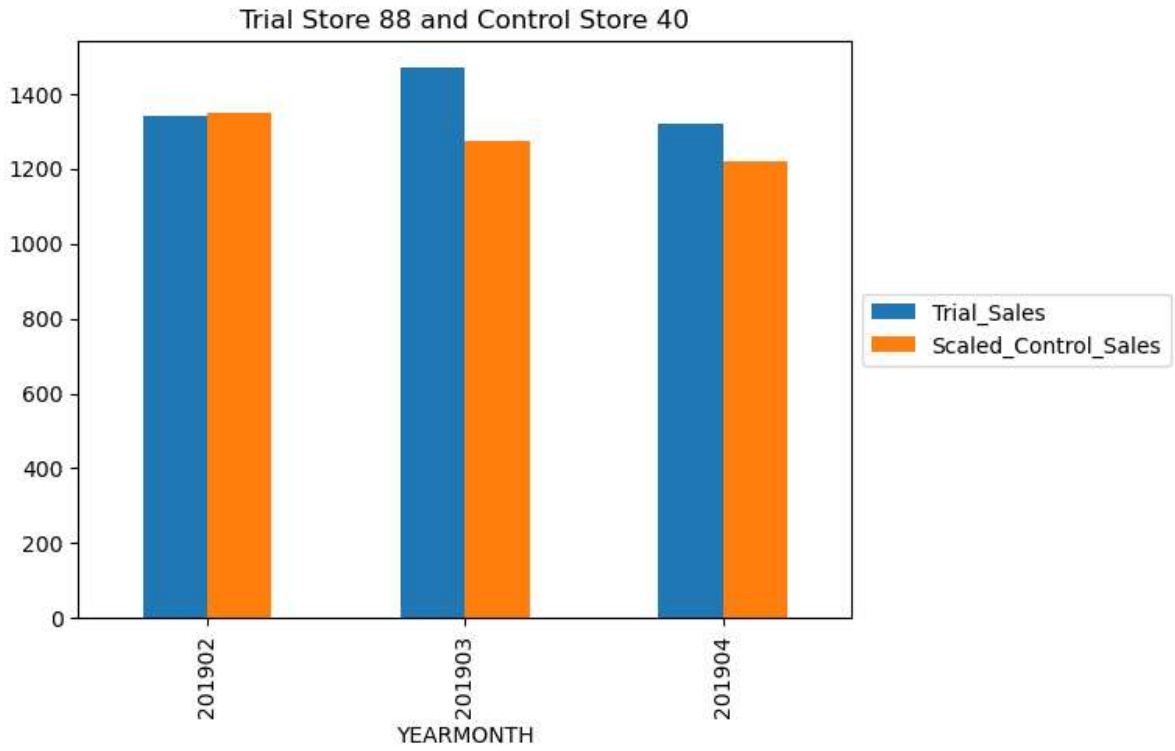
Next we'll compare the performance of Trial stores to Control stores during the trial period. To ensure their performance is comparable during Trial period, we need to scale (multiply to

ratio of trial / control) all of Control stores' performance to Trial store's performance during pre-trial. Starting with TOT_SALES.

In [23]:
```
#Ratio of Store 77 and its Control store.
sales_ratio_77 = preTrialMeasures[preTrialMeasures["STORE_NBR"] == 77]["TOT_SALES"

#Ratio of Store 86 and its Control store.
sales_ratio_86 = preTrialMeasures[preTrialMeasures["STORE_NBR"] == 86]["TOT_SALES"

#Ratio of Store 77 and its Control store.
sales_ratio_88 = preTrialMeasures[preTrialMeasures["STORE_NBR"] == 88]["TOT_SALES"
```

In [24]:
```
trial_full_observ = full_observ[(full_observ["YEARMONTH"] >= 201902) & (full_obser
scaled_sales_control_stores = full_observ[full_observ["STORE_NBR"].isin([233, 155,

def scaler(row):
    if row["STORE_NBR"] == 233:
        return row["TOT_SALES"] * sales_ratio_77
    elif row["STORE_NBR"] == 155:
        return row["TOT_SALES"] * sales_ratio_86
    elif row["STORE_NBR"] == 40:
        return row["TOT_SALES"] * sales_ratio_88

scaled_sales_control_stores["ScaledSales"] = scaled_sales_control_stores.apply(lam

trial_scaled_sales_control_stores = scaled_sales_control_stores[(scaled_sales_contr
pretrial_scaled_sales_control_stores = scaled_sales_control_stores[scaled_sales_co

percentage_diff = {}

for trial, control in trial_control_dic.items():
    a = trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["STORE_
    b = trial_full_observ[trial_full_observ["STORE_NBR"] == trial][["STORE_NBR", "
    percentage_diff[trial] = b["TOT_SALES"].sum() / a["ScaledSales"].sum()
    b[["YEARMONTH", "TOT_SALES"]].merge(a[["YEARMONTH", "ScaledSales"]],on="YEARMO
    plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
    plt.title("Trial Store "+str(trial)+" and Control Store "+str(control))
```





In [25]:
```
percentage_diff
```

Out[25]:
```
{77: 1.2778901246373204, 86: 1.1155403148404528, 88: 1.0736785754013871}
```

In [26]:
```
temp1 = scaled_sales_control_stores.sort_values(by=["STORE_NBR", "YEARMONTH"], asc
temp2 = full_observ[full_observ["STORE_NBR"].isin([77,86,88])][["STORE_NBR", "YEAR
scaledsales_vs_trial = pd.concat([temp1, temp2], axis=1)
scaledsales_vs_trial.columns = ["c_STORE_NBR", "YEARMONTH", "c_ScaledSales", "t_ST
scaledsales_vs_trial["Sales_Percentage_Diff"] = (scaledsales_vs_trial["t_TOT_SALES
def label_period(cell):
    if cell < 201902:
        return "pre"
    elif cell > 201904:
        return "post"
    else:
```

```
        return "trial"
scaledsales_vs_trial["trial_period"] = scaledsales_vs_trial["YEARMONTH"].apply(lam
scaledsales_vs_trial[scaledsales_vs_trial["trial_period"] == "trial"]
```

Out[26]:

	c_STORE_NBR	YEARMONTH	c_ScaledSales	t_STORE_NBR	t_TOT_SALES	Sales_Percentage_Diff
7	233	201902	229.473346	77	211.6	-0.081045
8	233	201903	187.779277	77	255.1	0.304014
9	233	201904	149.932291	77	258.1	0.530192
19	155	201902	827.019610	86	872.8	0.053865
20	155	201903	745.561872	86	945.4	0.236360
21	155	201904	778.028321	86	804.0	0.032833
31	40	201902	1346.153002	88	1339.6	-0.004880
32	40	201903	1273.550526	88	1467.0	0.141176
33	40	201904	1220.924472	88	1317.0	0.075712

Check significance of Trial minus Control stores TOT_SALES Percentage Difference Pre-Trial vs Trial.

Step 1: Check null hypothesis of 0 difference between control store's Pre-Trial and Trial period performance.

Step 2: Proof control and trial stores are similar statistically

Check p-value of control store's Pre-Trial vs Trial store's Pre-Trial. If <5%, it is significantly different. If >5%, it is not significantly different (similar).

Step 3: After checking Null Hypothesis of first 2 step to be true, we can check Null Hypothesis of percentage difference between Trial and Control stores during pre-trial is the same as during trial.

Check T-Value of Percentage Difference of each Trial month (Feb, March, April 2019). Mean is mean of Percentage Difference during pre-trial. Standard deviation is stdev of Percentage Difference during pre-trial. Formula is Trial month's Percentage Difference minus Mean, divided by Standard deviation. Compare each T-Value with 95% percentage significance critical t-value of 6 degrees of freedom (7 months of sample - 1)

Step 1: Check null hypothesis of 0 difference between control store's Pre-Trial and Trial period performance.

In [27]:

```
from scipy.stats import ttest_ind, t

# Step 1
for num in [40, 155, 233]:
    print("Store", num)
    print(ttest_ind(pretrial_scaled_sales_control_stores[pretrial_scaled_sales_cont
                    trial_scaled_sales_control_stores[trial_scaled_sales_control_st
                    equal_var=False), '\n')
    #print(len(pretrial_scaled_sales_control_stores[pretrial_scaled_sales_control_s
```

```
alpha = 0.05
print("Critical t-value for 95% confidence interval:")
print(t.ppf((alpha/2, 1-alpha/2), df=min([len(pretrial_scaled_sales_control_stores
                        len(trial_scaled_sales_control_stores[trial_scaled_sales_co
```

```
Store 40
Ttest_indResult(statistic=-0.36013226673707055, pvalue=0.7312292961250694)

Store 155
Ttest_indResult(statistic=1.2632008050177177, pvalue=0.24959862186836165)

Store 233
Ttest_indResult(statistic=1.2851941918470577, pvalue=0.2540387701484951)

Critical t-value for 95% confidence interval:
[-4.30265273  4.30265273]
```

In [28]:
```
a = pretrial_scaled_sales_control_stores[pretrial_scaled_sales_control_stores["STO
b = trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["STORE_NBR"
```

Null hypothesis is true. There is not any statistically significant difference between control store's scaled Pre-Trial and Trial period sales.

Step 2: Proof control and trial stores are similar statistically

In [29]:
```
# Step 2
for trial, cont in trial_control_dic.items():
    print("Trial store:", trial, ", Control store:", cont)
    print(ttest_ind(preTrialMeasures[preTrialMeasures["STORE_NBR"] == trial]["TOT_S
                    pretrial_scaled_sales_control_stores[pretrial_scaled_sales_contr
                    equal_var=True), '\n')
    #print(len(pretrial_full_observ[pretrial_full_observ["STORE_NBR"] == trial]["TO

alpha = 0.05
print("Critical t-value for 95% confidence interval:")
print(t.ppf((alpha/2, 1-alpha/2), df=len(preTrialMeasures[preTrialMeasures["STORE_N
```

```
Trial store: 77 , Control store: 233
Ttest_indResult(statistic=2.4926005499695936e-15, pvalue=0.999999999999998)

Trial store: 86 , Control store: 155
Ttest_indResult(statistic=0.0, pvalue=1.0)

Trial store: 88 , Control store: 40
Ttest_indResult(statistic=0.0, pvalue=1.0)

Critical t-value for 95% confidence interval:
[-2.44691185  2.44691185]
```

The null hypothesis holds true, indicating that there is no statistically significant distinction between the sales performance of the Trial store and the scaled sales performance of the Control store during the pre-trial period.

Step 3: Check Null Hypothesis of percentage difference between Trial and Control stores during pre-trial is the same as during trial.

In [30]:
```
# Step 3
for trial, cont in trial_control_dic.items():
    print("Trial store:", trial, ", Control store:", cont)
    temp_pre = scaledsales_vs_trial[(scaledsales_vs_trial["c_STORE_NBR"] == cont) &
```

```
    std = temp_pre["Sales_Percentage_Diff"].std()
    mean = temp_pre["Sales_Percentage_Diff"].mean()
    #print(std, mean)
    for t_month in scaledsales_vs_trial[scaledsales_vs_trial["trial_period"] == "t
        pdif = scaledsales_vs_trial[(scaledsales_vs_trial["YEARMONTH"] == t_month)
        print(t_month,":",(float(pdif)-mean)/std)
    print('\n')

print("Critical t-value for 95% confidence interval:")
conf_intv_95 = t.ppf(0.95, df=len(temp_pre)-1)
print(conf_intv_95)
```

```
Trial store: 77 , Control store: 233
201902 : -0.8435806220494326
201903 : 2.4904464734755805
201904 : 4.448806995302806


Trial store: 86 , Control store: 155
201902 : 1.3077596199049661
201903 : 5.828582596615282
201904 : 0.7867590051021528


Trial store: 88 , Control store: 40
201902 : -0.06747939544187861
201903 : 1.4413280336576249
201904 : 0.7650631287827934


Critical t-value for 95% confidence interval:
1.9431802803927816
```

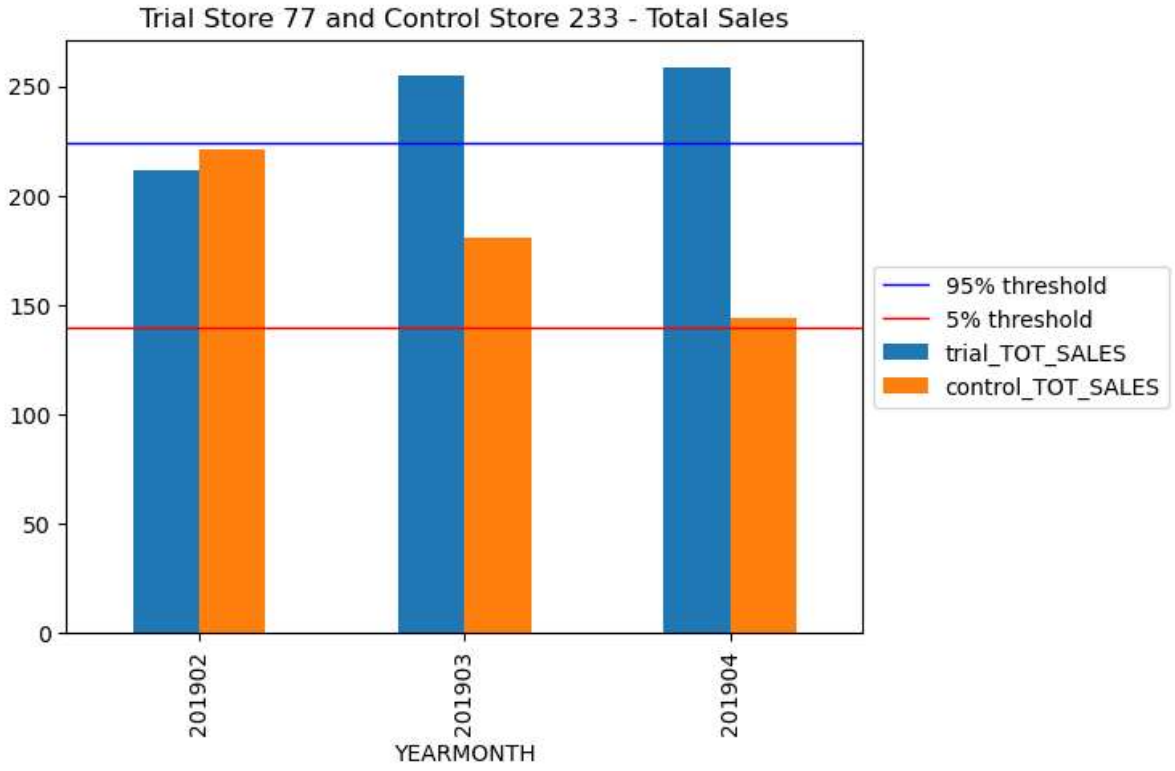
There are 3 months' increase in performance that are statistically significant (Above the 95% confidence interval t-score):

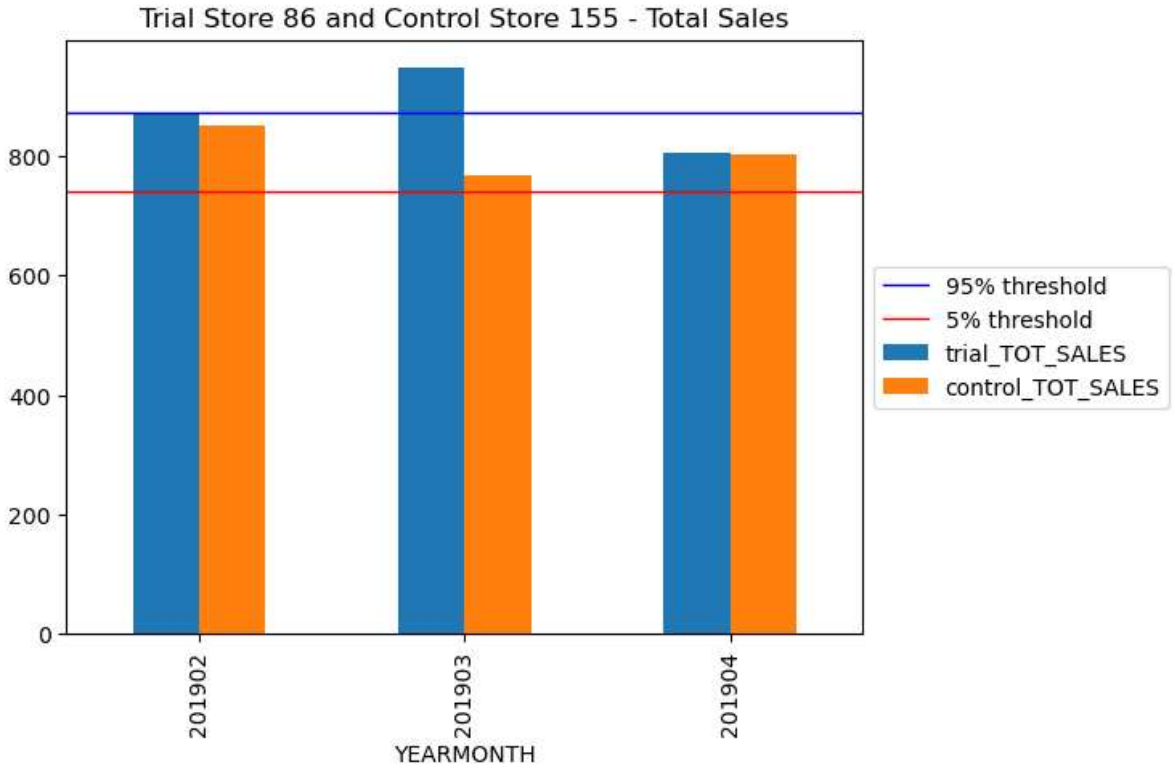
March and April trial months for trial store 77

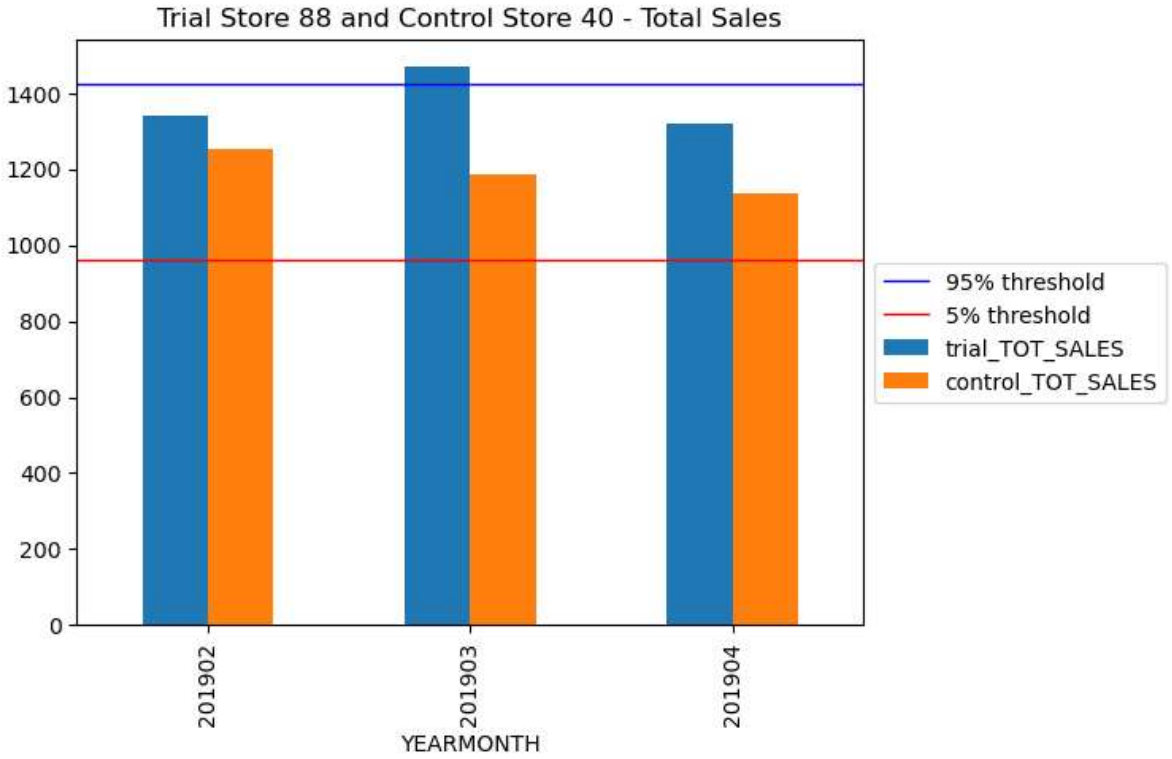
March trial months for trial store 86

In [ ]:

In [31]:
```
for trial, control in trial_control_dic.items():
    a = trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["STORE_
    b = trial_full_observ[trial_full_observ["STORE_NBR"] == trial][["STORE_NBR", "
    comb = b[["YEARMONTH", "trial_TOT_SALES"]].merge(a[["YEARMONTH", "control_TOT_
    comb.plot.bar()
    cont_sc_sales = trial_scaled_sales_control_stores[trial_scaled_sales_control_st
    std = scaledsales_vs_trial[(scaledsales_vs_trial["c_STORE_NBR"] == control) &
    thresh95 = cont_sc_sales.mean() + (cont_sc_sales.mean() * std * 2)
    thresh5 = cont_sc_sales.mean() - (cont_sc_sales.mean() * std * 2)
    plt.axhline(y=thresh95,linewidth=1, color='b', label="95% threshold")
    plt.axhline(y=thresh5,linewidth=1, color='r', label="5% threshold")
    plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
    plt.title("Trial Store "+str(trial)+" and Control Store "+str(control)+" - Tota
    plt.savefig("TS {} and CS {} - TOT_SALES.png".format(trial,control), bbox_inche
```

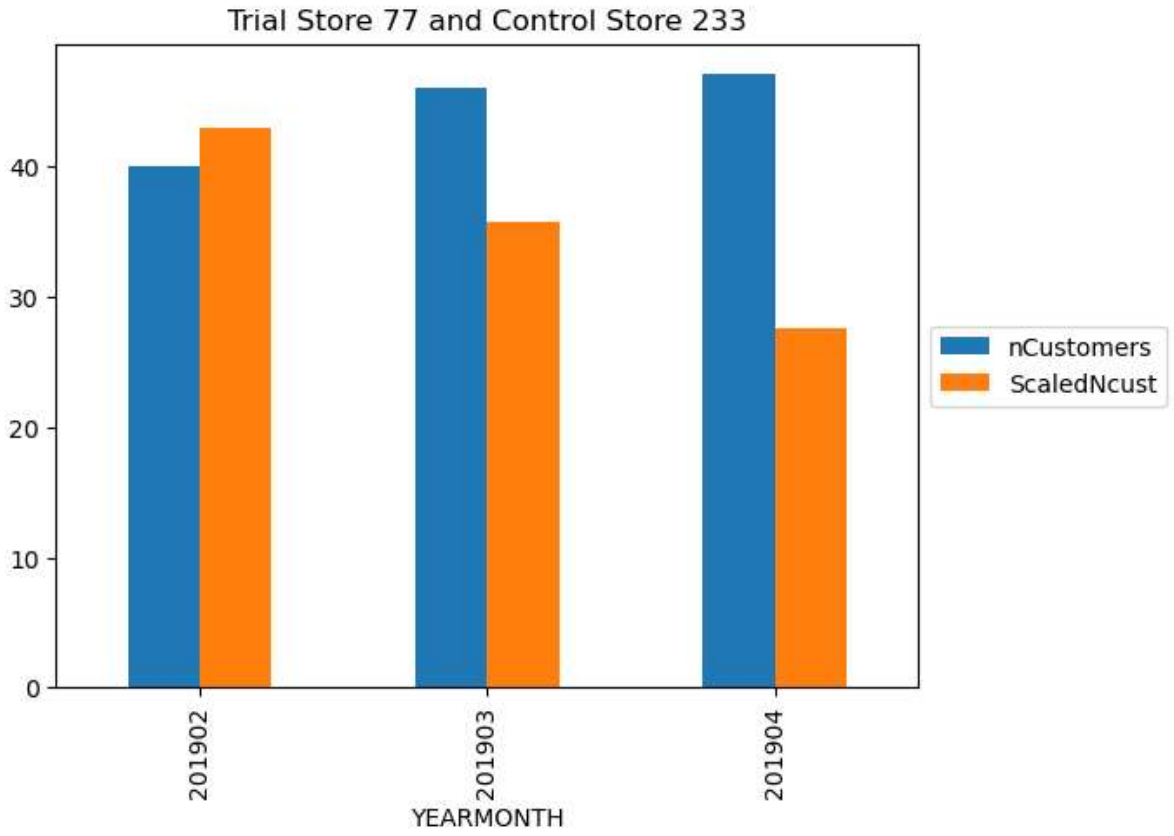

Trial Store 77 and Control Store 233 - Total Sales
250
200
150
100
50
0
201902
201903
201904
YEARMONTH
95% threshold
5% threshold
trial_TOT_SALES
control_TOT_SALES

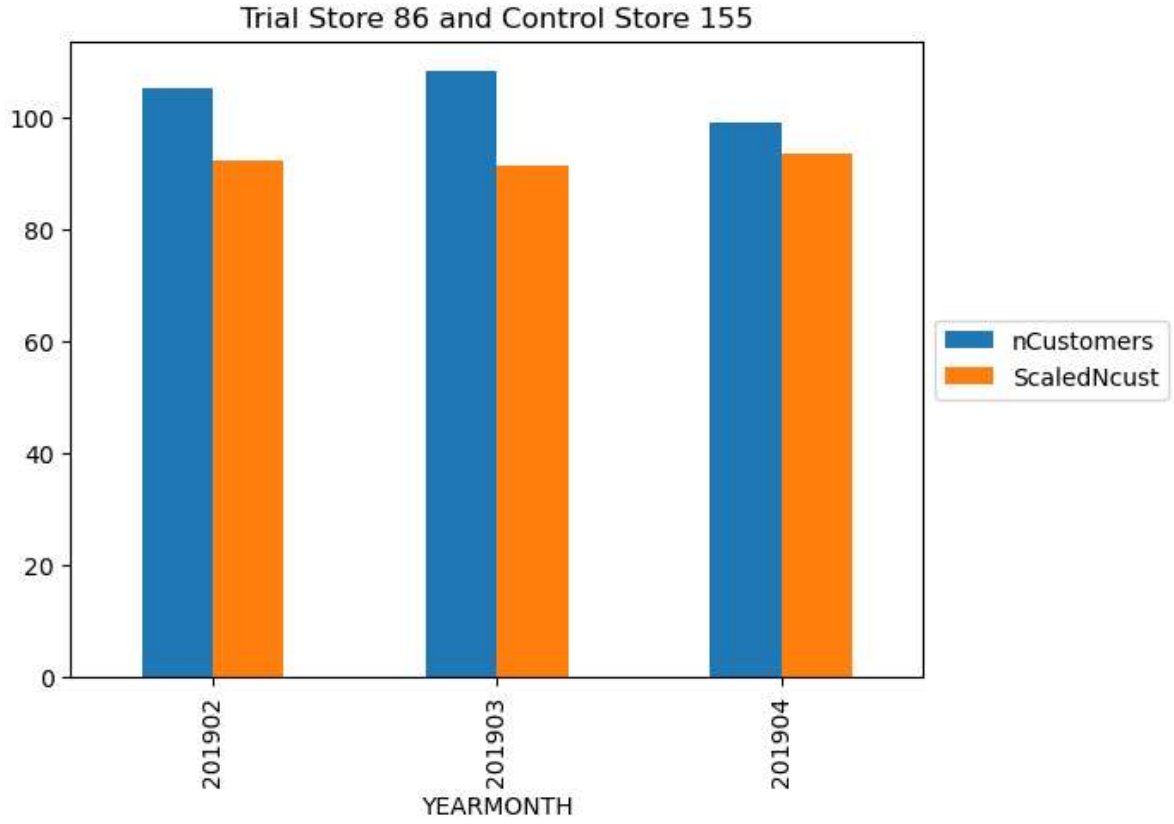

Trial Store 86 and Control Store 155 - Total Sales
800
600
400
200
0
201902
201903
201904
YEARMONTH
95% threshold
5% threshold
trial_TOT_SALES
control_TOT_SALES

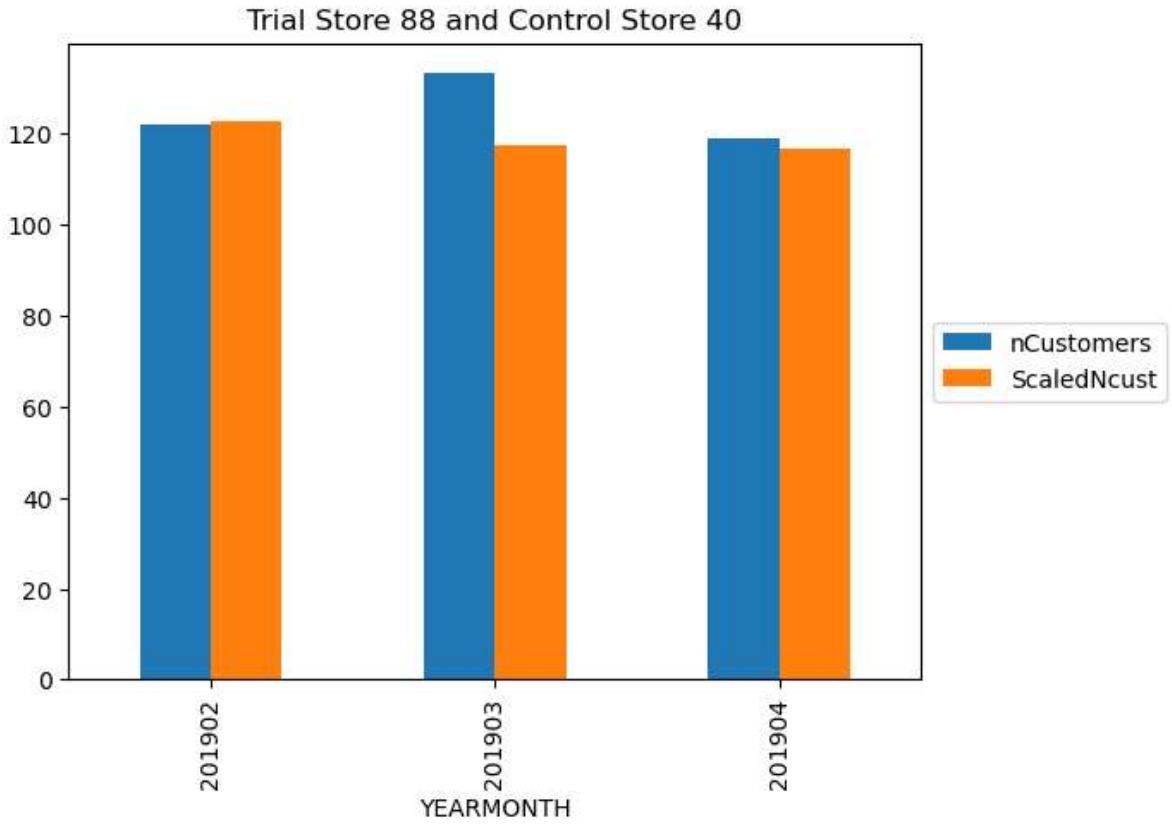


In [32]:
```
#Ratio of Store 77 and its Control store.
ncust_ratio_77 = preTrialMeasures[preTrialMeasures["STORE_NBR"] == 77]["nCustomers

#Ratio of Store 86 and its Control store.
ncust_ratio_86 = preTrialMeasures[preTrialMeasures["STORE_NBR"] == 86]["nCustomers

#Ratio of Store 77 and its Control store.
ncust_ratio_88 = preTrialMeasures[preTrialMeasures["STORE_NBR"] == 88]["nCustomers
```

In [33]:
```
#trial_full_observ = full_observ[(full_observ["YEARMONTH"] >= 201902) & (full_obser
scaled_ncust_control_stores = full_observ[full_observ["STORE_NBR"].isin([233, 155,

def scaler_c(row):
    if row["STORE_NBR"] == 233:
        return row["nCustomers"] * ncust_ratio_77
    elif row["STORE_NBR"] == 155:
        return row["nCustomers"] * ncust_ratio_86
    elif row["STORE_NBR"] == 40:
        return row["nCustomers"] * ncust_ratio_88

scaled_ncust_control_stores["ScaledNcust"] = scaled_ncust_control_stores.apply(lam

trial_scaled_ncust_control_stores = scaled_ncust_control_stores[(scaled_ncust_cont
pretrial_scaled_ncust_control_stores = scaled_ncust_control_stores[scaled_ncust_co

ncust_percentage_diff = {}

for trial, control in trial_control_dic.items():
    a = trial_scaled_ncust_control_stores[trial_scaled_ncust_control_stores["STORE_
    b = trial_full_observ[trial_full_observ["STORE_NBR"] == trial][["STORE_NBR", "
    ncust_percentage_diff[trial] = b["nCustomers"].sum() / a["ScaledNcust"].sum()
    b[["YEARMONTH", "nCustomers"]].merge(a[["YEARMONTH", "ScaledNcust"]],on="YEARM
    plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
    plt.title("Trial Store "+str(trial)+" and Control Store "+str(control))
```


Trial Store 77 and Control Store 233
40
30
20
10
0
201902
201903
201904
YEARMONTH
nCustomers
ScaledNcust


Trial Store 86 and Control Store 155
100
80
60
40
20
0
201902
201903
201904
YEARMONTH
nCustomers
ScaledNcust



In [34]:
```
ncust_percentage_diff
```

Out[34]:
```
{77: 1.2516366612111292, 86: 1.1270502473314241, 88: 1.0492326310578137}
```

In [35]:
```
temp1 = scaled_ncust_control_stores.sort_values(by=["STORE_NBR", "YEARMONTH"], asc
temp2 = full_observ[full_observ["STORE_NBR"].isin([77,86,88])][["STORE_NBR", "YEAR
scaledncust_vs_trial = pd.concat([temp1, temp2], axis=1)
scaledncust_vs_trial.columns = ["c_STORE_NBR", "YEARMONTH", "c_ScaledNcust", "t_ST
scaledncust_vs_trial["nCust_Percentage_Diff"] = (scaledncust_vs_trial["t_nCustomer

scaledncust_vs_trial["trial_period"] = scaledncust_vs_trial["YEARMONTH"].apply(lam
scaledncust_vs_trial[scaledncust_vs_trial["trial_period"] == "trial"]
```

Out[35]:

	c_STORE_NBR	YEARMONTH	c_ScaledNcust	t_STORE_NBR	t_nCustomers	nCust_Percentage_Di
7	233	201902	42.913043	77	40	-0.07026
8	233	201903	35.760870	77	46	0.25046
9	233	201904	27.586957	77	47	0.52054
19	155	201902	92.276276	86	105	0.12899
20	155	201903	91.273273	86	108	0.16787
21	155	201904	93.279279	86	99	0.05950
31	40	201902	122.627451	88	122	-0.00513
32	40	201903	117.431373	88	133	0.12433
33	40	201904	116.392157	88	119	0.02215

In [ ]:
```
```

Check significance of Trial minus Control stores nCustomers Percentage Difference Pre-Trial vs Trial.

Step 1: Check null hypothesis of 0 difference between control store's Pre-Trial and Trial period performance.

Step 2: Proof control and trial stores are similar statistically

Step 3: After checking Null Hypothesis of first 2 step to be true, we can check Null Hypothesis of Percentage Difference between Trial and Control stores during pre-trial is the same as during trial.

Step 1: Check null hypothesis of 0 difference between control store's Pre-Trial and Trial period performance.

In [36]:
```
for num in [40, 155, 233]:
    print("Store", num)
    print(ttest_ind(pretrial_scaled_ncust_control_stores[pretrial_scaled_ncust_cont
                    trial_scaled_ncust_control_stores[trial_scaled_ncust_control_st
                    equal_var=False), '\n')

alpha = 0.05
print("Critical t-value for 95% confidence interval:")
print(t.ppf((alpha/2, 1-alpha/2), df=min([len(pretrial_scaled_ncust_control_stores
                        len(trial_scaled_ncust_control_stores[trial_scaled_ncust_co
```

```
Store 40
Ttest_indResult(statistic=0.5635718480314612, pvalue=0.5885700785723794)

Store 155
Ttest_indResult(statistic=1.4443707614569483, pvalue=0.19292354514578497)

Store 233
Ttest_indResult(statistic=0.9728013089400613, pvalue=0.39878209711761925)

Critical t-value for 95% confidence interval:
[-4.30265273  4.30265273]
```

In [ ]:
```
```

Step 2: Proof control and trial stores are similar statistically

In [37]:
```
for trial, cont in trial_control_dic.items():
    print("Trial store:", trial, ", Control store:", cont)
    print(ttest_ind(preTrialMeasures[preTrialMeasures["STORE_NBR"] == trial]["nCust
                    pretrial_scaled_ncust_control_stores[pretrial_scaled_ncust_contr
                    equal_var=True), '\n')

alpha = 0.05
print("Critical t-value for 95% confidence interval:")
print(t.ppf((alpha/2, 1-alpha/2), df=len(preTrialMeasures[preTrialMeasures["STORE_
```

```
Trial store: 77 , Control store: 233
Ttest_indResult(statistic=0.0, pvalue=1.0)

Trial store: 86 , Control store: 155
Ttest_indResult(statistic=0.0, pvalue=1.0)

Trial store: 88 , Control store: 40
Ttest_indResult(statistic=0.0, pvalue=1.0)

Critical t-value for 95% confidence interval:
[-2.44691185  2.44691185]
```

In [ ]:

Step 3: Check Null Hypothesis of Percentage Difference between Trial and Control stores during pre-trial is the same as during trial.

In [38]:
```
for trial, cont in trial_control_dic.items():
    print("Trial store:", trial, ", Control store:", cont)
    temp_pre = scaledncust_vs_trial[(scaledncust_vs_trial["c_STORE_NBR"] == cont) &
    std = temp_pre["nCust_Percentage_Diff"].std()
    mean = temp_pre["nCust_Percentage_Diff"].mean()
    #print(std, mean)
    for t_month in scaledncust_vs_trial[scaledncust_vs_trial["trial_period"] == "tr
        pdif = scaledncust_vs_trial[(scaledncust_vs_trial["YEARMONTH"] == t_month)
        print(t_month,":",(float(pdif)-mean)/std)
    print('\n')

print("Critical t-value for 95% confidence interval:")
conf_intv_95 = t.ppf(0.95, df=len(temp_pre)-1)
print(conf_intv_95)
```

```
Trial store: 77 , Control store: 233
201902 : -1.6033778348003613
201903 : 5.5462661353472695
201904 : 11.566838770938542


Trial store: 86 , Control store: 155
201902 : 3.775892411526406
201903 : 4.914998319676522
201904 : 1.7401582372094115


Trial store: 88 , Control store: 40
201902 : -0.08952991281963425
201903 : 1.4499020670055427
201904 : 0.2349371961466534


Critical t-value for 95% confidence interval:
1.9431802803927816
```

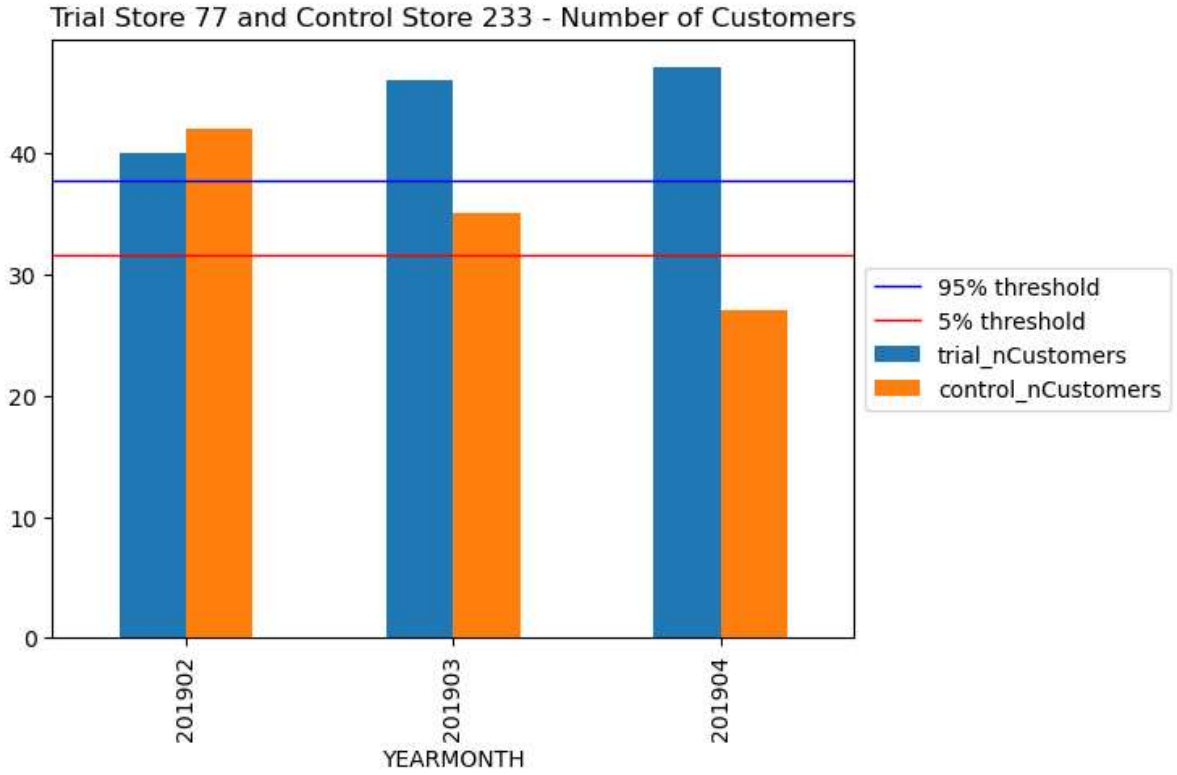
There are 5 months' increase in performance that are statistically significant (Above the 95% confidence interval t-score):

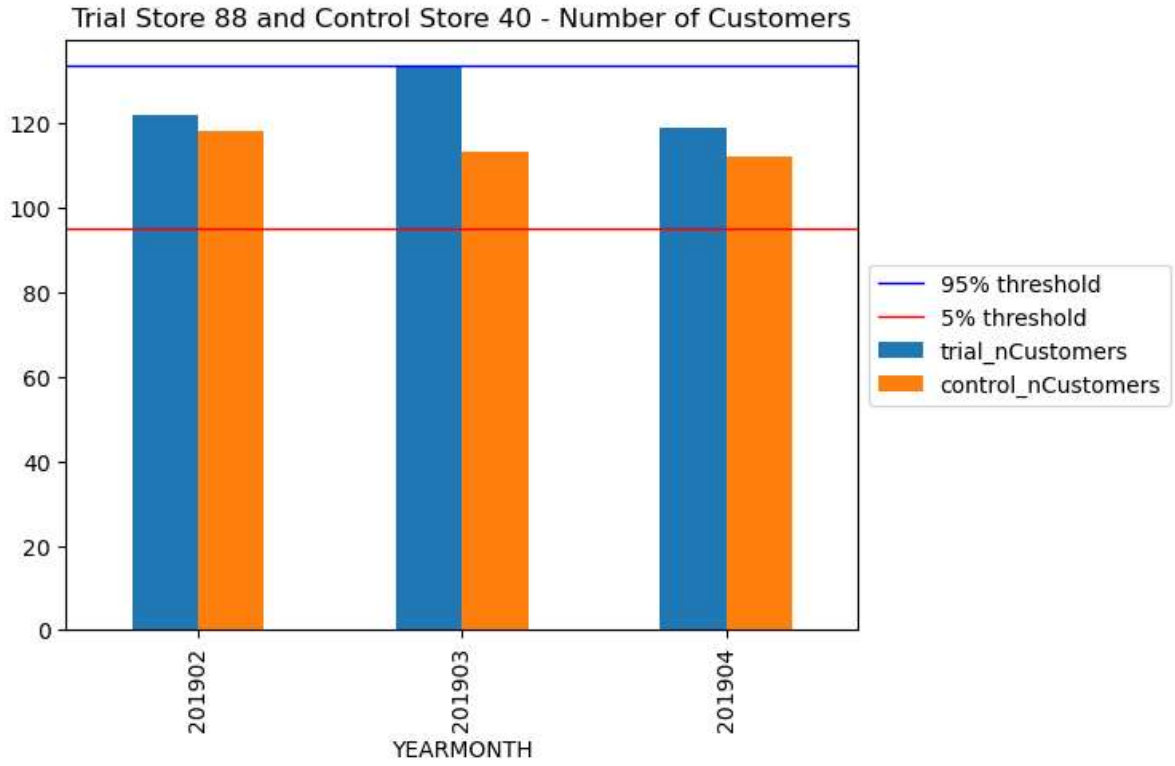
March and April trial months for trial store 77

Feb, March and April trial months for trial store 86

In [39]:
```
for trial, control in trial_control_dic.items():
    a = trial_scaled_ncust_control_stores[trial_scaled_ncust_control_stores["STORE_
```

```
    b = trial_full_observ[trial_full_observ["STORE_NBR"] == trial][["STORE_NBR", "Y
    comb = b[["YEARMONTH", "trial_nCustomers"]].merge(a[["YEARMONTH", "control_nCus
    comb.plot.bar()
    cont_sc_ncust = trial_scaled_ncust_control_stores[trial_scaled_ncust_control_st
    std = scaledncust_vs_trial[(scaledncust_vs_trial["c_STORE_NBR"] == control) &
    thresh95 = cont_sc_ncust.mean() + (cont_sc_ncust.mean() * std * 2)
    thresh5 = cont_sc_ncust.mean() - (cont_sc_ncust.mean() * std * 2)
    plt.axhline(y=thresh95,linewidth=1, color='b', label="95% threshold")
    plt.axhline(y=thresh5,linewidth=1, color='r', label="5% threshold")
    plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
    plt.title("Trial Store "+str(trial)+" and Control Store "+str(control)+" - Numl
    plt.savefig("TS {} and CS {} - nCustomers.png".format(trial,control), bbox_incl
```





In [ ]:

We can see that Trial store 77 sales for Feb, March, and April exceeds 95% threshold of control store. Same goes to store 86 sales for all 3 trial months.

1. Trial store 77: Control store 233
2. Trial store 86: Control store 155
3. Trial store 88: Control store 40

The analysis reveals that Trial stores 77 and 86 exhibited a significant increase in total sales and number of customers during the three-month trial period, exceeding the 95% threshold of their respective Control stores. Conversely, Trial store 88 did not exhibit a significant increase in performance during the trial period compared to its Control store. It is possible that there were certain factors that distinguished Trial store 88 from the other two Trial stores, which could have contributed to this difference in performance. However, on the whole, the trial showed a positive and statistically significant outcome.

In [ ]:

In [ ]: