

Name - Ashlesh Khajbage

Task 2 :-

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.
 - This can be broken down by:-
 - ◆ Total sales revenue
 - ◆ Total number of customers
 - ◆ 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 :-
 - Select control stores – Explore data, define metrics, visualize graphs
 - Assessment of the trial – insights/trends by comparing trial stores with control stores
 - Collate findings – summarize and provide recommendations

Importing Necessary Libraries

```
In [1]: import pandas as pd
import numpy as np

# for data visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Importing Dataset

```
In [3]: qvi = pd.read_csv("Data/QVI_data.csv")
qvi.head()
```

Out[3]:

	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	
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	
3	1003	2019-03-08	1	4	106	Natural ChipCo Hony Soy Chckn175g	1	
4	1004	2018-11-02	1	5	96	WW Original Stacked Chips 160g	1	

Data Exploration

In [4]:

```
print("Number of Rows and Columns :- ", qvi.shape)
```

Number of Rows and Columns :- (264834, 12)

In [5]:

```
# Basic Information of dataset
qvi.info()
```

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

In [6]:

```
# Statistical Summary of QVI_data
qvi.describe().T
```

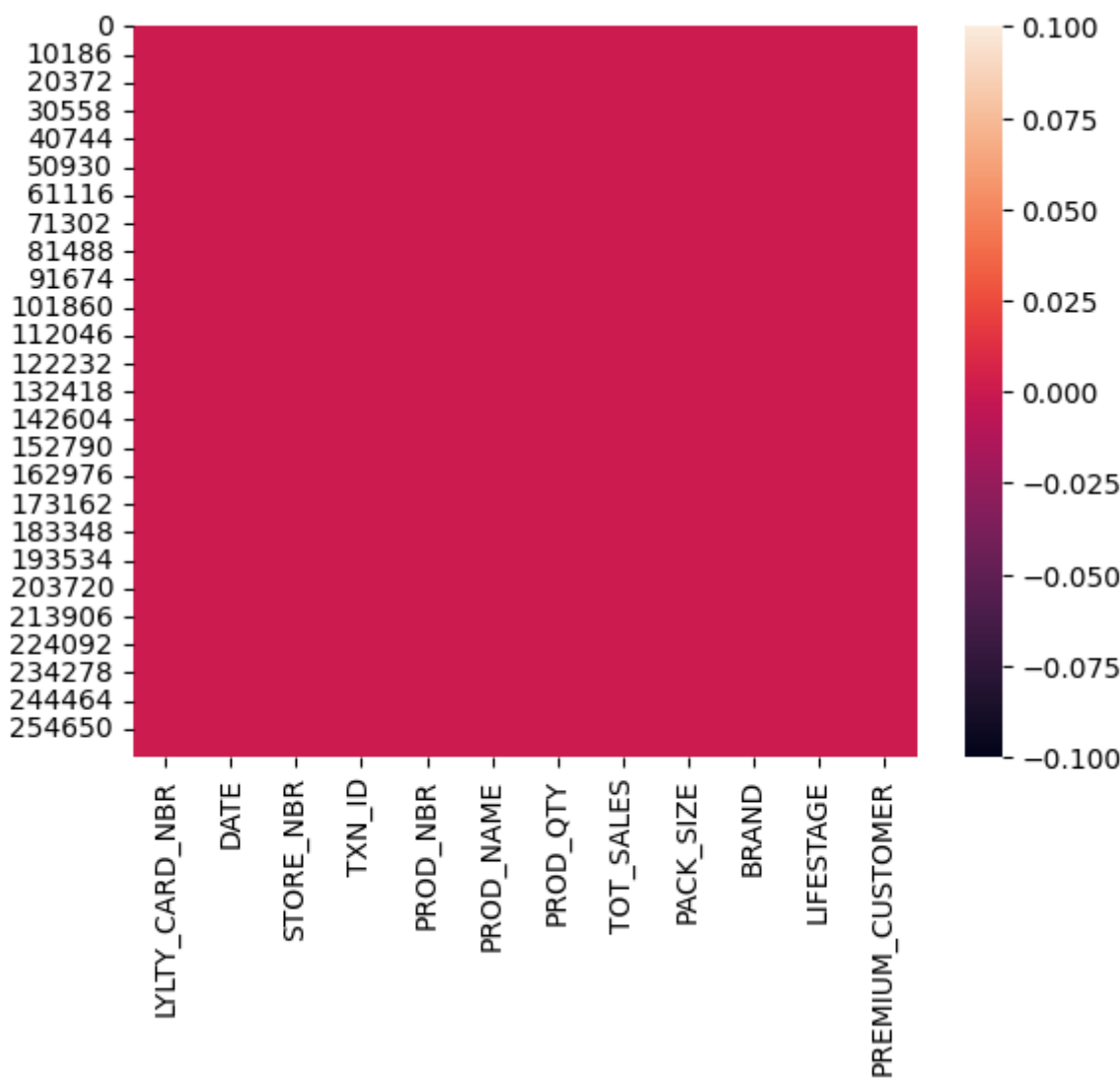
Out [6]:

	count	mean	std	min	25%	50%	75%
LYLTY_CARD_NBR	264834.0	135548.793331	80579.898912	1000.0	70021.0	130357.0	203094.0
STORE_NBR	264834.0	135.079423	76.784063	1.0	70.0	130.0	203.0
TXN_ID	264834.0	135157.623236	78132.920436	1.0	67600.5	135136.5	202699.7
PROD_NBR	264834.0	56.583554	32.826444	1.0	28.0	56.0	85.0
PROD_QTY	264834.0	1.905813	0.343436	1.0	2.0	2.0	2.0
TOT_SALES	264834.0	7.299346	2.527241	1.5	5.4	7.4	9.2
PACK_SIZE	264834.0	182.425512	64.325148	70.0	150.0	170.0	175.0

Checking missing values in Dataset

In [7]:

```
sns.heatmap(qvi.isnull())  
plt.show()
```



In [8]:

```
qvi.isnull().sum()
```

```
Out[8]: LYLTY_CARD_NBR      0
        DATE              0
        STORE_NBR        0
        TXN_ID            0
        PROD_NBR          0
        PROD_NAME         0
        PROD_QTY          0
        TOT_SALES         0
        PACK_SIZE         0
        BRAND             0
        LIFESTAGE          0
        PREMIUM_CUSTOMER  0
        dtype: int64
```

We can see there is no missing values the dataset.

```
In [9]: ### Handling "Date" column
qvi["DATE"] = pd.to_datetime(qvi["DATE"])
qvi["YEARMONTH"] = qvi["DATE"].dt.strftime("%Y%m").astype("int")
```

Compile each store's monthly:-

- Total sales
- Number of customers
- Average transactions per customer
- Average chips per customer
- Average price per unit

```
In [11]: def monthly_store_metrics():
        store_yrmo_group = qvi.groupby(["STORE_NBR", "YEARMONTH"])
        total = store_yrmo_group["TOT_SALES"].sum()
        num_cust = store_yrmo_group["LYLTY_CARD_NBR"].nunique()
        trans_per_cust = store_yrmo_group.size() / num_cust
        avg_chips_per_cust = store_yrmo_group["PROD_QTY"].sum() / num_cust
        avg_chips_price = total / store_yrmo_group["PROD_QTY"].sum()
        aggregates = [total, num_cust, trans_per_cust, avg_chips_per_cust, avg_chips_price]
        metrics = pd.concat(aggregates, axis=1)
        metrics.columns = ["TOT_SALES", "nCustomers", "nTxnPerCust", "nChipsPerTxn", "avgPricePerUnit"]
        return metrics
```

```
In [12]: qvi_monthly_metrics = monthly_store_metrics().reset_index()
qvi_monthly_metrics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3169 entries, 0 to 3168
Data columns (total 7 columns):
#   Column              Non-Null Count  Dtype
---  -
0   STORE_NBR           3169 non-null  int64
1   YEARMONTH           3169 non-null  int64
2   TOT_SALES           3169 non-null  float64
3   nCustomers          3169 non-null  int64
4   nTxnPerCust         3169 non-null  float64
5   nChipsPerTxn        3169 non-null  float64
6   avgPricePerUnit     3169 non-null  float64
dtypes: float64(4), int64(3)
memory usage: 173.4 KB
```

Pre-Trial Observation as this filter only stores with full 12 months observation

```
In [14]: observ_counts = qvi_monthly_metrics["STORE_NBR"].value_counts()
full_observ_index = observ_counts[observ_counts == 12].index
full_observ = qvi_monthly_metrics[qvi_monthly_metrics["STORE_NBR"].isin(full_observ_index)]
pretrial_full_observ = full_observ[full_observ["YEARMONTH"] < 201902]

pretrial_full_observ.head(8)
```

```
Out[14]:
```

	STORE_NBR	YEARMONTH	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerChip
0	1	201807	206.9	49	1.061224	1.265306	3.0
1	1	201808	176.1	42	1.023810	1.285714	3.0
2	1	201809	278.8	59	1.050847	1.271186	3.0
3	1	201810	188.1	44	1.022727	1.318182	3.0
4	1	201811	192.6	46	1.021739	1.239130	3.0
5	1	201812	189.6	42	1.119048	1.357143	3.0
6	1	201901	154.8	35	1.028571	1.200000	3.0
12	2	201807	150.8	39	1.051282	1.179487	3.0

```
In [15]: def calcCorrTable(metricCol, storeComparison, inputTable=pretrial_full_observ):
control_store_nbrs = inputTable[~inputTable["STORE_NBR"].isin([77, 86, 88])]
corrs = pd.DataFrame(columns = ["YEARMONTH", "Trial_Str", "Ctrl_Str", "Corr_Score"])
trial_store = inputTable[inputTable["STORE_NBR"] == storeComparison][metricCol]
for control in control_store_nbrs:
concat_df = pd.DataFrame(columns = ["YEARMONTH", "Trial_Str", "Ctrl_Str", "Corr_Score"])
control_store = inputTable[inputTable["STORE_NBR"] == control][metricCol]
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"] == storeComparison]["YEARMONTH"])
corrs = pd.concat([corrs, concat_df])
return corrs
```

```
In [16]: corr_table = pd.DataFrame()
for trial_num in [77, 86, 88]:
corr_table = pd.concat([corr_table, calcCorrTable(["TOT_SALES", "nCustomers", "nChipsPerTxn", "avgPricePerChip"], trial_num, inputTable)])
corr_table.head(8)
```

Out[16]:

	YEARMONTH	Trial_Str	Ctrl_Str	Corr_Score
0	201807	77	1	0.070414
1	201808	77	1	0.027276
2	201809	77	1	0.002389
3	201810	77	1	-0.020045
4	201811	77	1	0.030024
5	201812	77	1	0.063946
6	201901	77	1	0.001470
0	201807	77	2	0.142957

```
In [17]: def calculateMagnitudeDistance(metricCol, storeComparison, inputTable=pretrial_
control_store_nbrs = inputTable[~inputTable["STORE_NBR"].isin([77, 86, 88])]
dists = pd.DataFrame()
trial_store = inputTable[inputTable["STORE_NBR"] == storeComparison][metricCol]
for control in control_store_nbrs:
    concat_df = abs(inputTable[inputTable["STORE_NBR"] == storeComparison]
    concat_df["YEARMONTH"] = list(inputTable[inputTable["STORE_NBR"] == storeComparison])
    concat_df["Trial_Str"] = storeComparison
    concat_df["Ctrl_Str"] = control
    dists = pd.concat([dists, concat_df])
for col in metricCol:
    dists[col] = 1 - ((dists[col] - dists[col].min()) / (dists[col].max() - dists[col].min()))
dists["magnitude"] = dists[metricCol].mean(axis=1)
return dists
```

```
In [18]: dist_table = pd.DataFrame()
for trial_num in [77, 86, 88]:
    dist_table = pd.concat([dist_table, calculateMagnitudeDistance(["TOT_SALES", "YEARMONTH", "Trial_Str", "Ctrl_Str"], trial_num, inputTable)])
dist_table.head(8)
dist_table
```

Out[18]:

	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	YEARMONTH	Trial_
0	0.935431	0.980769	0.958035	0.739412	0.883569	201807	
1	0.942972	0.951923	0.993823	0.802894	0.886328	201808	
2	0.961503	0.836538	0.992126	0.730041	0.703027	201809	
3	0.988221	0.932692	0.989514	0.940460	0.590528	201810	
4	0.962149	0.951923	0.874566	0.730358	0.832481	201811	
...
2	0.207554	0.286822	0.462846	0.779879	0.923887	201809	
3	0.346797	0.387597	0.571497	0.796875	0.971133	201810	
4	0.286706	0.310078	0.623883	0.813241	0.966999	201811	
5	0.347151	0.387597	0.376456	0.699748	0.962198	201812	
6	0.402353	0.449612	0.450378	0.739714	0.971335	201901	

5397 rows × 9 columns

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 [23]: def combine_corr_dist(metricCol, storeComparison, inputTable=pretrial_full_obs):
    corrs = calcCorrTable(metricCol, storeComparison, inputTable)
    dists = calculateMagnitudeDistance(metricCol, storeComparison, inputTable)
    dists = dists.drop(metricCol, axis=1)
    combine = pd.merge(corrs, dists, on=["YEARMONTH", "Trial_Str", "Ctrl_Str"])
    return combine
```

```
In [24]: compare_metrics_table1 = pd.DataFrame()
    for trial_num in [77, 86, 88]:
        compare_metrics_table1 = pd.concat([compare_metrics_table1, combine_corr_di
```

```
In [25]: corr_weight = 0.5
    dist_weight = 1 - corr_weight
```

Determining the top five highest composite score for each trial based on Total sales

```
In [26]: grouped_comparison_table1 = compare_metrics_table1.groupby(["Trial_Str", "Ctrl_
    grouped_comparison_table1["CompScore"] = (corr_weight * grouped_comparison_tabl
    for trial_num in compare_metrics_table1["Trial_Str"].unique():
        print(grouped_comparison_table1[grouped_comparison_table1["Trial_Str"] == t
```

	Trial_Str	Ctrl_Str	Corr_Score	magnitude	CompScore
218	77	233	1.0	0.986477	0.993238
239	77	255	1.0	0.979479	0.989739
177	77	188	1.0	0.977663	0.988831
49	77	53	1.0	0.976678	0.988339
120	77	131	1.0	0.976267	0.988134

	Trial_Str	Ctrl_Str	Corr_Score	magnitude	CompScore
356	86	109	1.0	0.966783	0.983391
401	86	155	1.0	0.965876	0.982938
464	86	222	1.0	0.962280	0.981140
467	86	225	1.0	0.960512	0.980256
471	86	229	1.0	0.951704	0.975852

	Trial_Str	Ctrl_Str	Corr_Score	magnitude	CompScore
551	88	40	1.0	0.941165	0.970582
538	88	26	1.0	0.904377	0.952189
582	88	72	1.0	0.903800	0.951900
517	88	4	1.0	0.903466	0.951733
568	88	58	1.0	0.891678	0.945839

```
In [27]: compare_metrics_table2 = pd.DataFrame()
for trial_num in [77, 86, 88]:
    compare_metrics_table2 = pd.concat([compare_metrics_table2, combine_corr_di
```

Determining the top five highest composite score for each trial based on no. of customers

```
In [28]: grouped_comparison_table2 = compare_metrics_table2.groupby(["Trial_Str", "Ctrl_
grouped_comparison_table2["CompScore"] = (corr_weight * grouped_comparison_tabl
for trial_num in compare_metrics_table2["Trial_Str"].unique():
    print(grouped_comparison_table2[grouped_comparison_table2["Trial_Str"] == t
```

	Trial_Str	Ctrl_Str	Corr_Score	magnitude	CompScore
218	77	233	1.0	0.993132	0.996566
38	77	41	1.0	0.976648	0.988324
101	77	111	1.0	0.968407	0.984203
105	77	115	1.0	0.967033	0.983516
15	77	17	1.0	0.965659	0.982830

	Trial_Str	Ctrl_Str	Corr_Score	magnitude	CompScore
401	86	155	1.0	0.986772	0.993386
467	86	225	1.0	0.969577	0.984788
356	86	109	1.0	0.969577	0.984788
471	86	229	1.0	0.964286	0.982143
293	86	39	1.0	0.961640	0.980820

	Trial_Str	Ctrl_Str	Corr_Score	magnitude	CompScore
736	88	237	1.0	0.987818	0.993909
705	88	203	1.0	0.944629	0.972315
551	88	40	1.0	0.942414	0.971207
668	88	165	1.0	0.935770	0.967885
701	88	199	1.0	0.932447	0.966224

```
In [29]: for trial_num in compare_metrics_table2["Trial_Str"].unique():
a = grouped_comparison_table1[grouped_comparison_table1["Trial_Str"] == tri
```



```

b = grouped_comparison_table2[grouped_comparison_table2["Trial_Str"] == tri
print((pd.concat([a,b], axis=1).sum(axis=1)/2).sort_values(ascending=False))

Trial_Str  Ctrl_Str
77          233      0.994902
           41      0.986020
           46      0.984762
dtype: float64

Trial_Str  Ctrl_Str
86          155      0.988162
           109      0.984090
           225      0.982522
dtype: float64

Trial_Str  Ctrl_Str
88          40      0.970895
           26      0.958929
           72      0.954079
dtype: float64

```

Observation

Similarities based on total sales:

1. Trial store 77: Store 233, 255, 188
2. Trial store 86: Store 109, 155, 222
3. Trial store 88: Store 40, 26, 72

Similarities based on No. of Customers:

1. Trial store 77: Store 233, 41, 111
2. Trial store 86: Store 155, 225, 109
3. Trial store 88: Store 237, 203, 40

Final Similarities based on Highest average of both features combined:

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

```

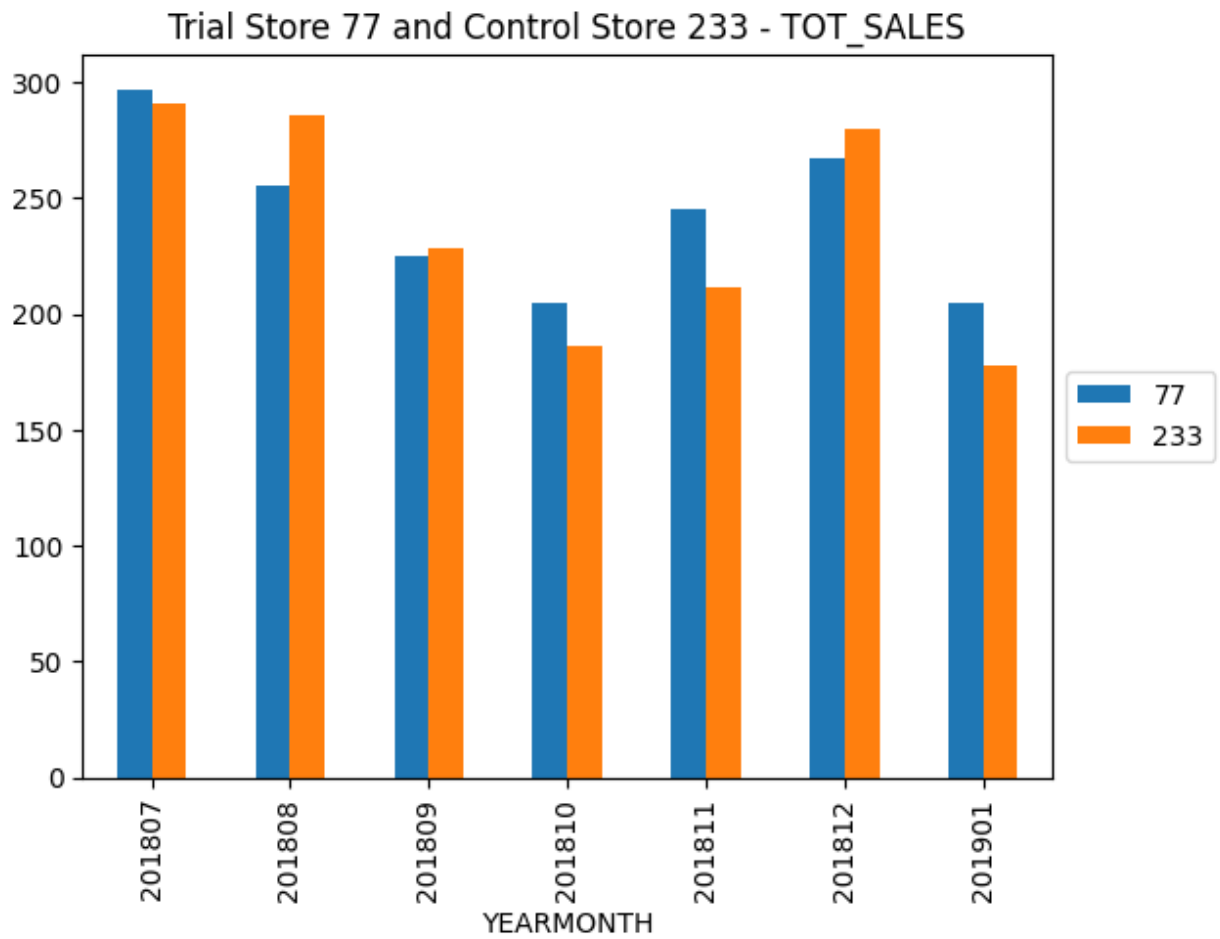
In [30]: trial_control_dic = {77:233, 86:155, 88:40}

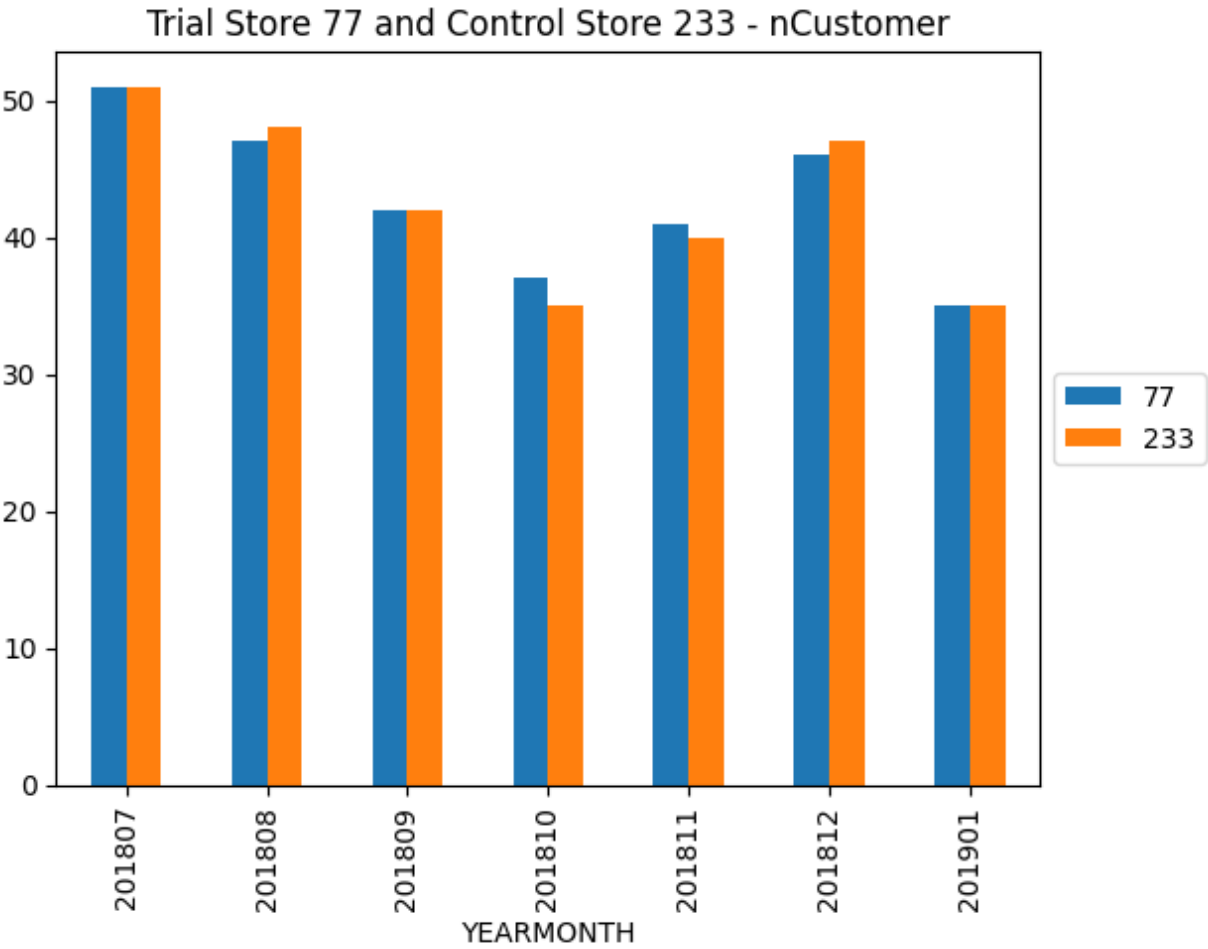
for key, val in trial_control_dic.items():
    pretrial_full_observ[pretrial_full_observ["STORE_NBR"].isin([key, val])]
    ["YEARMONTH", "STORE_NBR"].sum()["TOT_SALES"].unstack().plot.bar()
    plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
    plt.title("Trial Store "+str(key)+" and Control Store "+str(val)+" - TOT_SA
    plt.show()

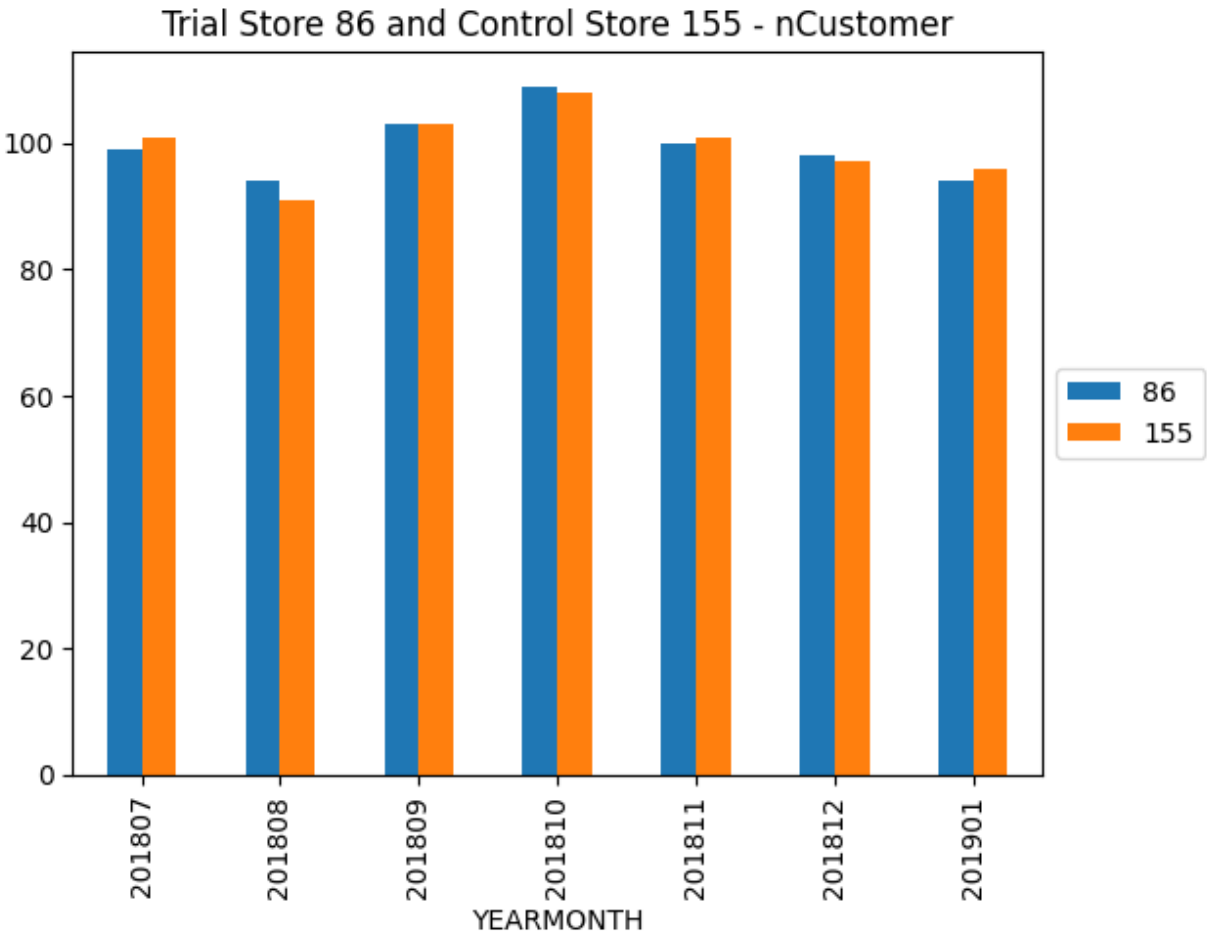
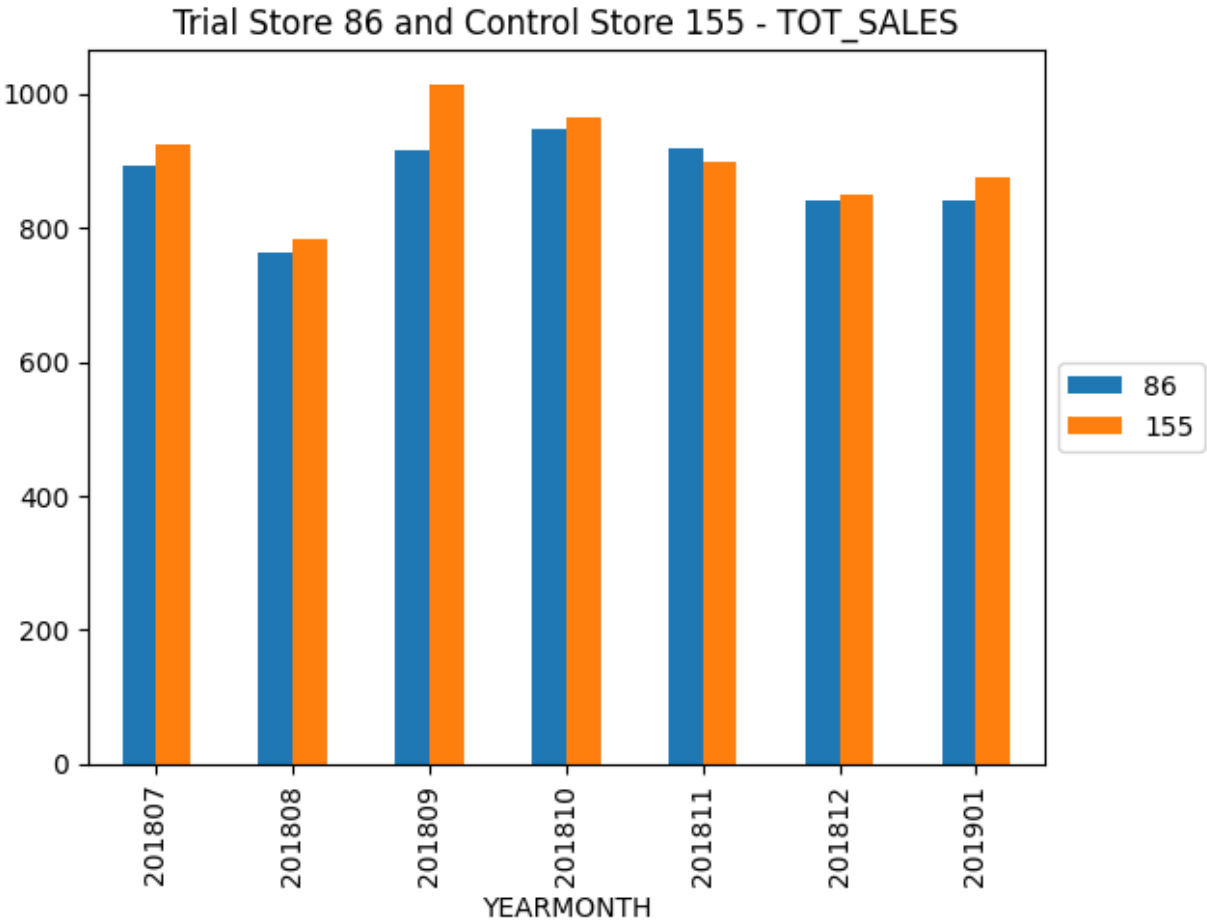
    pretrial_full_observ[pretrial_full_observ["STORE_NBR"].isin([key, val])]
    ["YEARMONTH", "STORE_NBR"].sum()["nCustomers"].unstack().plot.bar()
    plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
    plt.title("Trial Store "+str(key)+" and Control Store "+str(val)+" - nCusto

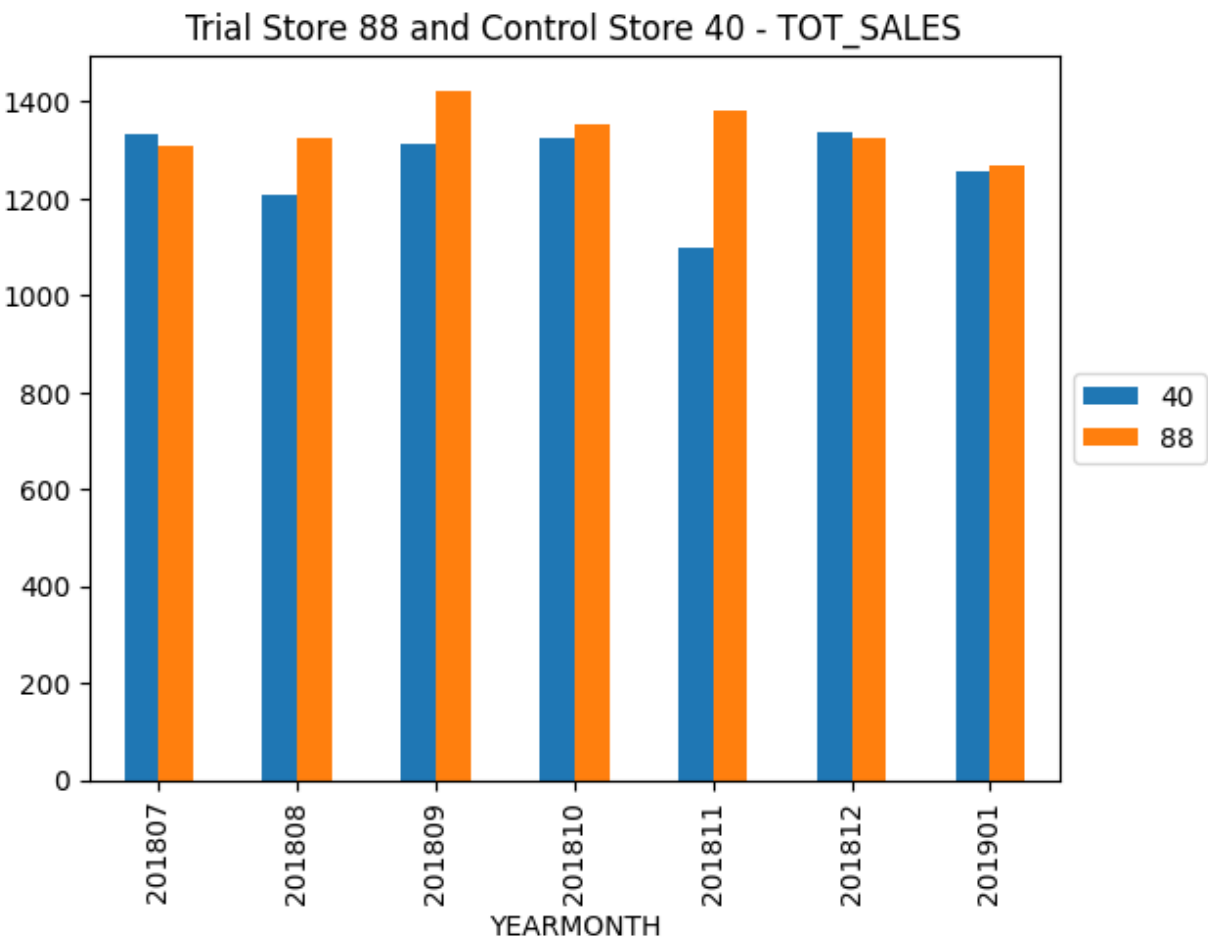
```

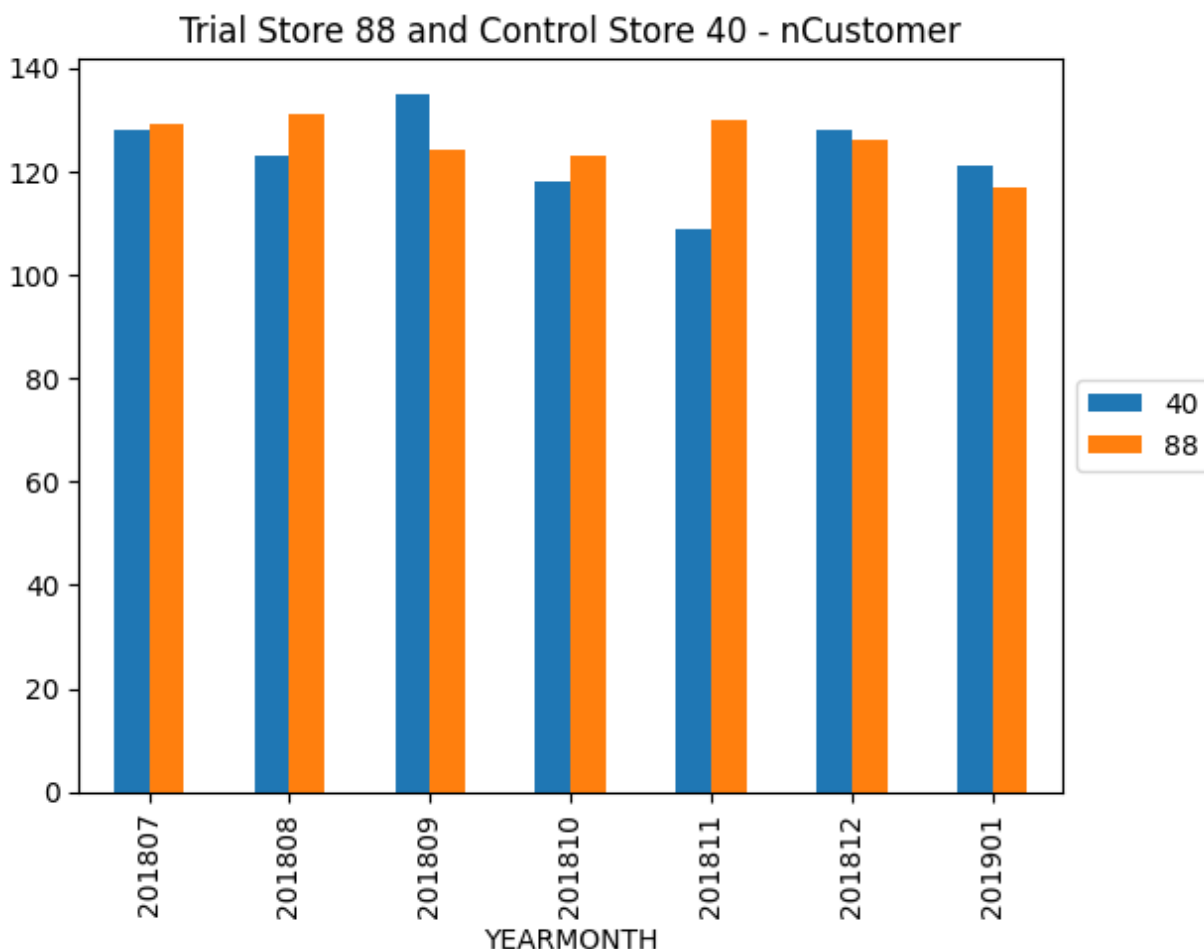
```
plt.show()  
print('\n')
```











Performance of Trails Store to Control Store during trial period

- 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 [31]: #Ratio of Store 77 and its Control store.
sales_ratio_77 = pretrial_full_observ[pretrial_full_observ["STORE_NBR"] == 77][

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

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

```
In [32]: trial_full_observ = full_observ[(full_observ["YEARMONTH"] >= 201902) & (full_observ["STORE_NBR"] != 88)]
scaled_sales_control_stores = full_observ[full_observ["STORE_NBR"].isin([233, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1620, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1720, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737, 1738, 1739, 1740, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1748, 1749, 1750, 1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1848, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891, 1892, 1893, 1894, 1895, 1896, 1897, 1898, 1899, 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166
```

```

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

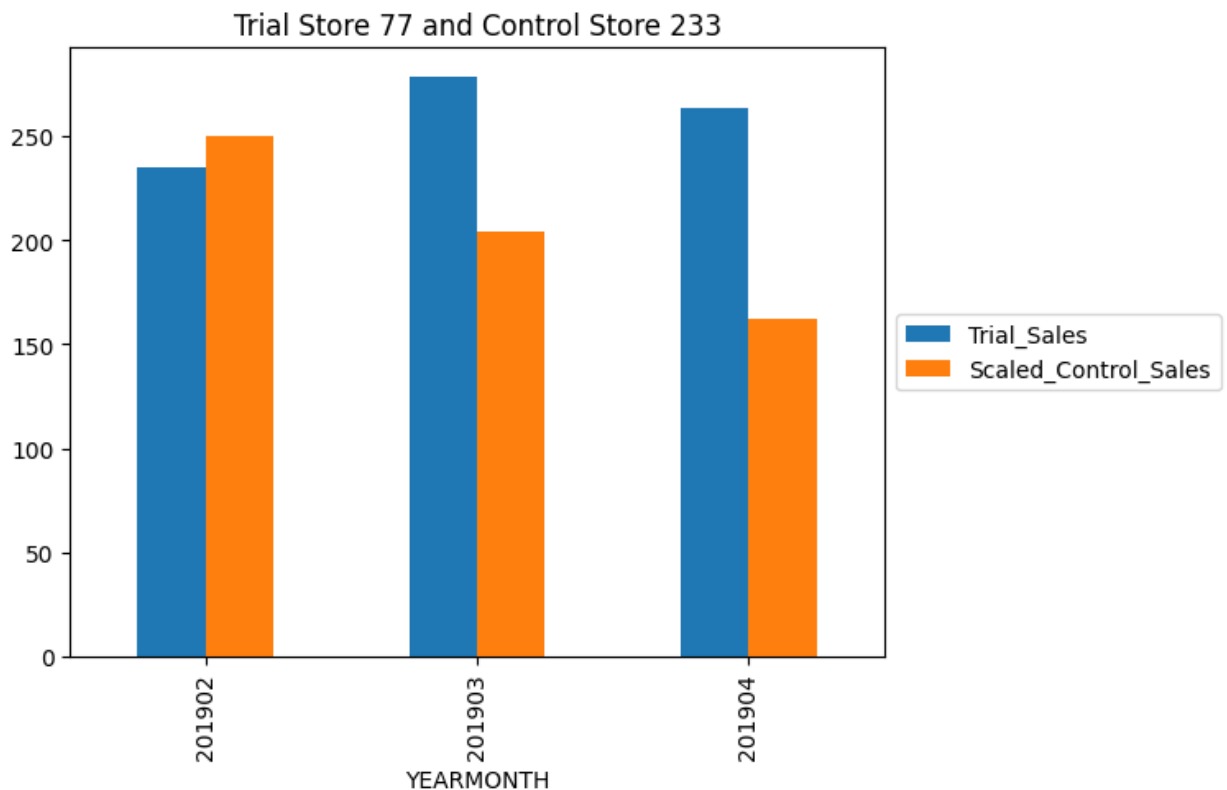
scaled_sales_control_stores["ScaledSales"] = scaled_sales_control_stores.apply(

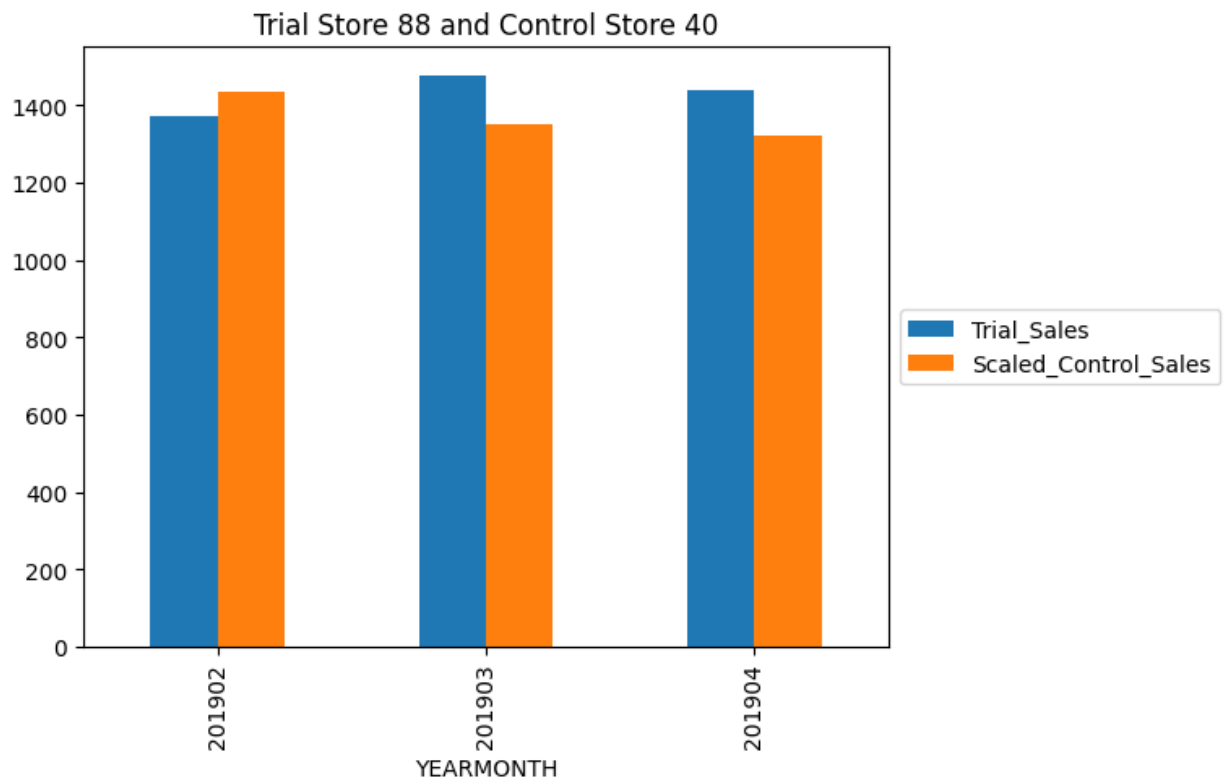
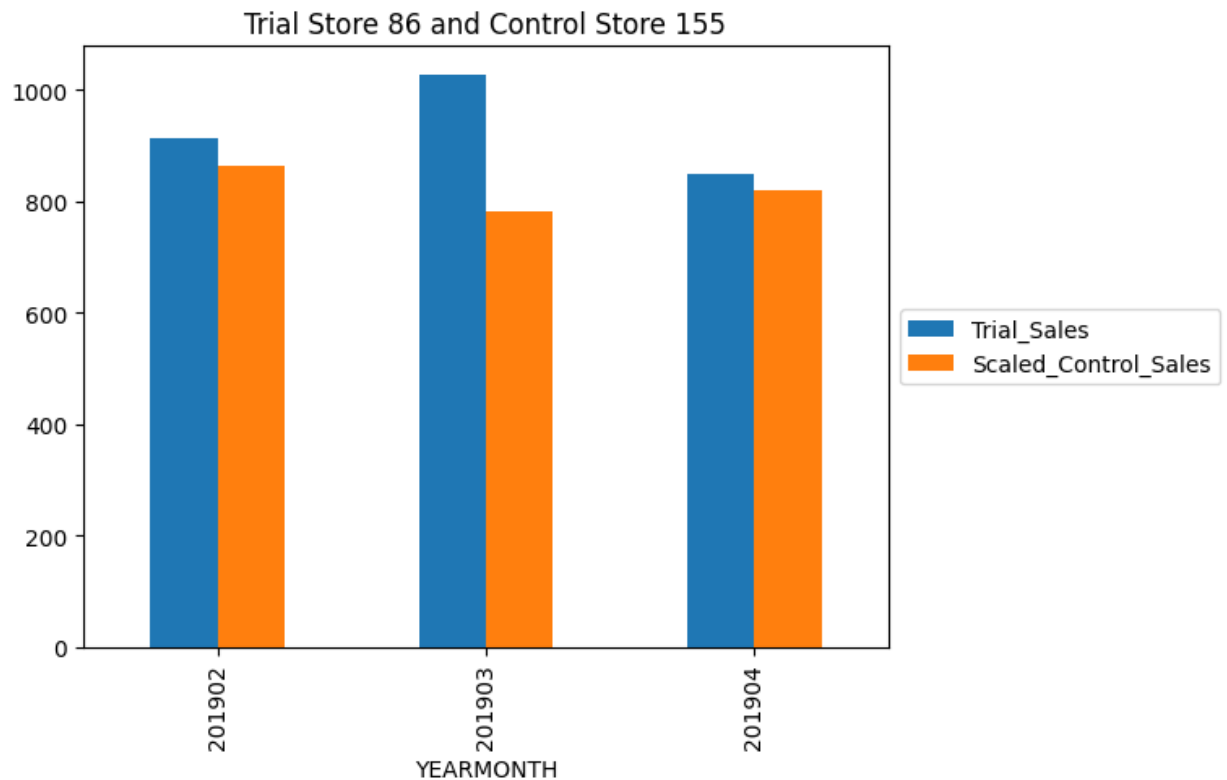
trial_scaled_sales_control_stores = scaled_sales_control_stores[(scaled_sales_c
pretrial_scaled_sales_control_stores = scaled_sales_control_stores[scaled_sales

percentage_diff = {}

for trial, control in trial_control_dic.items():
    a = trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["ST
    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="YEA
    plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
    plt.title("Trial Store "+str(trial)+" and Control Store "+str(control))

```





In [33]: `percentage_diff`

Out[33]: {77: 1.2615468650086281, 86: 1.1315014357363697, 88: 1.043458345854219}

```
In [34]: temp1 = scaled_sales_control_stores.sort_values(by=["STORE_NBR", "YEARMONTH"],
temp2 = full_observ[full_observ["STORE_NBR"].isin([77,86,88])][["STORE_NBR", "Y
scaledsales_vs_trial = pd.concat([temp1, temp2], axis=1)
scaledsales_vs_trial.columns = ["c_STORE_NBR", "YEARMONTH", "c_ScaledSales", "t
scaledsales_vs_trial["Sales_Percentage_Diff"] = (scaledsales_vs_trial["t_TOT_SF
```



```
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(
    lambda x: label_period(x) if x in [201902, 201903, 201904] else "trial")
```

Out[34]:

	c_STORE_NBR	YEARMONTH	c_ScaledSales	t_STORE_NBR	t_TOT_SALES	Sales_Percentage
7	233	201902	249.762622	77	235.0	-0.0
8	233	201903	203.802205	77	278.5	0.3
9	233	201904	162.345704	77	263.5	0.4
19	155	201902	864.522060	86	913.2	0.0
20	155	201903	780.320405	86	1026.8	0.2
21	155	201904	819.317024	86	848.2	0.0
31	40	201902	1434.399269	88	1370.2	-0.0
32	40	201903	1352.064709	88	1477.2	0.0
33	40	201904	1321.797762	88	1439.4	0.0

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)

```
In [35]: 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_
```

```

        trial_scaled_sales_control_stores[trial_scaled_sales_control
        equal_var=False), '\n')
    #print(len(pretrial_scaled_sales_control_stores[pretrial_scaled_sales_conti

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_sto
        len(trial_scaled_sales_control_stores[trial_scaled_sales

```

Store 40

Ttest_indResult(statistic=-0.5958372343168558, pvalue=0.5722861621434027)

Store 155

Ttest_indResult(statistic=1.4291956879290917, pvalue=0.1972705865160342)

Store 233

Ttest_indResult(statistic=1.191102601097452, pvalue=0.2944500606486209)

Critical t-value for 95% confidence interval:

[-4.30265273 4.30265273]

In [36]: a = pretrial_scaled_sales_control_stores[pretrial_scaled_sales_control_stores['
b = trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["STORE_

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

In [37]: # Step 2
for trial, cont in trial_control_dic.items():
 print("Trial store:", trial, ", Control store:", cont)
 print(ttest_ind(pretrial_full_observ[pretrial_full_observ["STORE_NBR"] == t
 pretrial_scaled_sales_control_stores[pretrial_scaled_sales_c
 equal_var=True), '\n')
 #print(len(pretrial_full_observ[pretrial_full_observ["STORE_NBR"] == trial]

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

Trial store: 77 , Control store: 233

Ttest_indResult(statistic=-1.2533353315065932e-15, pvalue=0.9999999999999999)

Trial store: 86 , Control store: 155

Ttest_indResult(statistic=3.1048311203382156e-15, pvalue=0.9999999999999976)

Trial store: 88 , Control store: 40

Ttest_indResult(statistic=-5.69358613974361e-15, pvalue=0.9999999999999956)

Critical t-value for 95% confidence interval:

[-2.44691185 2.44691185]

Null hypothesis is true. There isn't any statistically significant difference between Trial store's sales and Control store's scaled-sales performance during pre-trial.

In [38]: # 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"] == cor
 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"] ==
    pdif = scaledsales_vs_trial[(scaledsales_vs_trial["YEARMONTH"] == t_mon
    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 : -0.7171038288055838
 201903 : 3.035317928855674
 201904 : 4.708944418758219

Trial store: 86 , Control store: 155
 201902 : 1.4133618775921597
 201903 : 7.123063846042147
 201904 : 0.8863824572944234

Trial store: 88 , Control store: 40
 201902 : -0.5481633746817577
 201903 : 1.0089992743637823
 201904 : 0.9710006270463672

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):

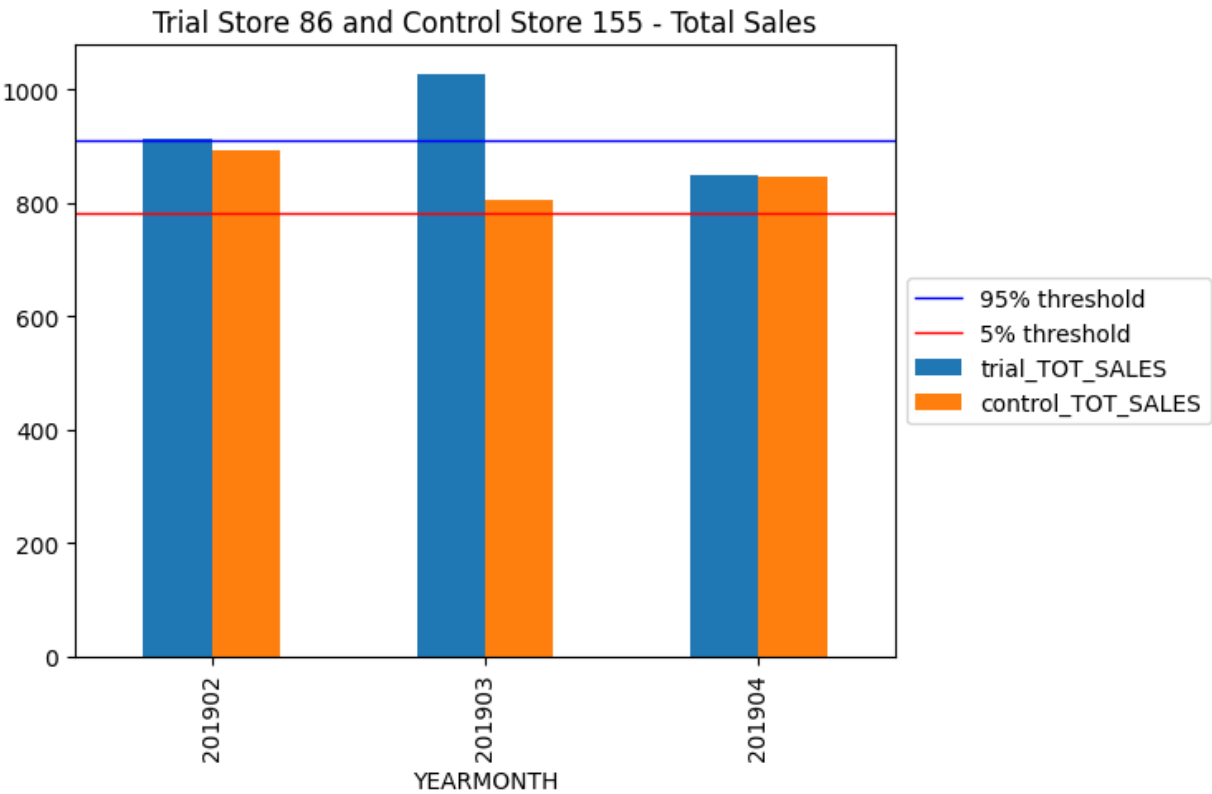
March and April trial months for trial store 77

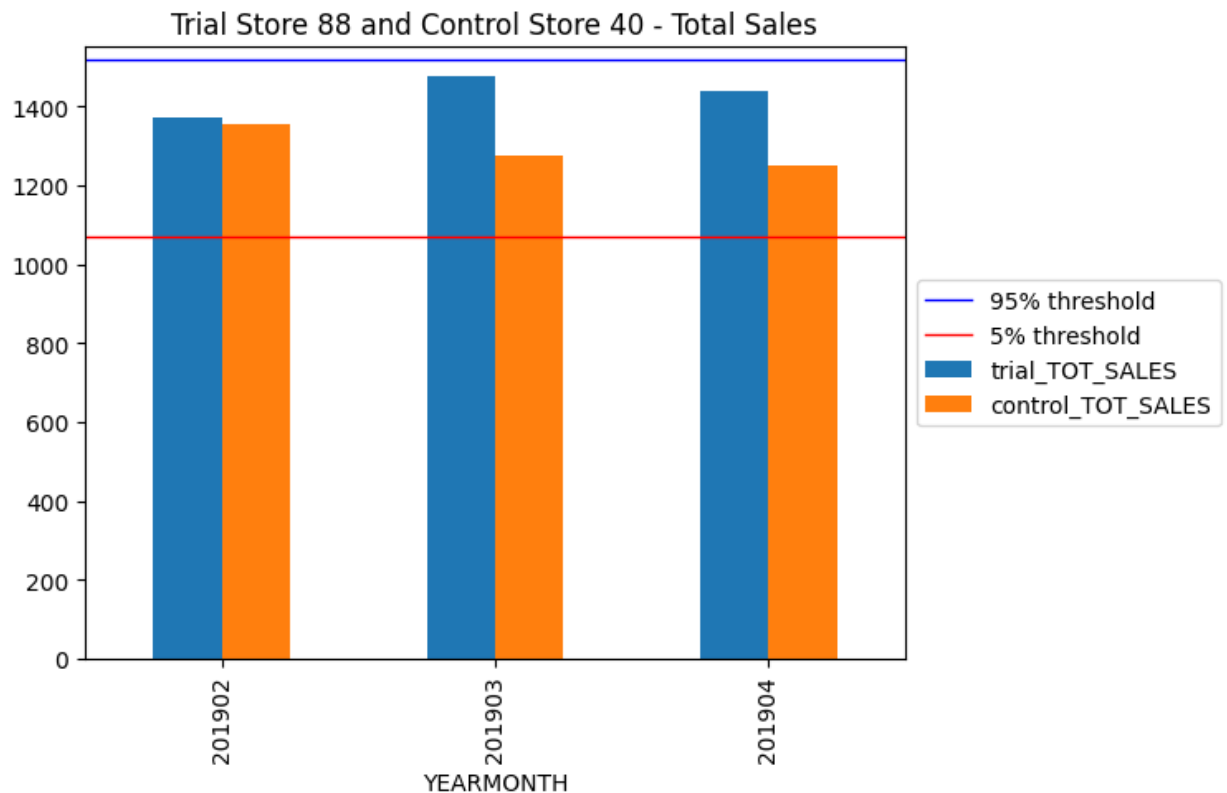
March trial months for trial store 86

```

In [39]: for trial, control in trial_control_dic.items():
    a = trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["ST
    b = trial_full_observ[trial_full_observ["STORE_NBR"] == trial][["STORE_NBR"
    comb = b[["YEARMONTH", "trial_TOT_SALES"]].merge(a[["YEARMONTH", "control_T
    comb.plot.bar()
    cont_sc_sales = trial_scaled_sales_control_stores[trial_scaled_sales_control
    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)+" -
    plt.savefig("TS {} and CS {} - TOT_SALES.png".format(trial,control), bbox_i

```





```
In [40]: #Ratio of Store 77 and its Control store.
ncust_ratio_77 = pretrial_full_observ[pretrial_full_observ["STORE_NBR"] == 77][
    "nCustomers"] / pretrial_full_observ[pretrial_full_observ["STORE_NBR"] == 77][
    "nCustomers"]

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

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

```
In [41]: #trial_full_observ = full_observ[(full_observ["YEARMONTH"] >= 201902) & (full_observ["STORE_NBR"]
scaled_ncust_control_stores = full_observ[full_observ["STORE_NBR"].isin([233, 155, 40])][
    "nCustomers"]

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

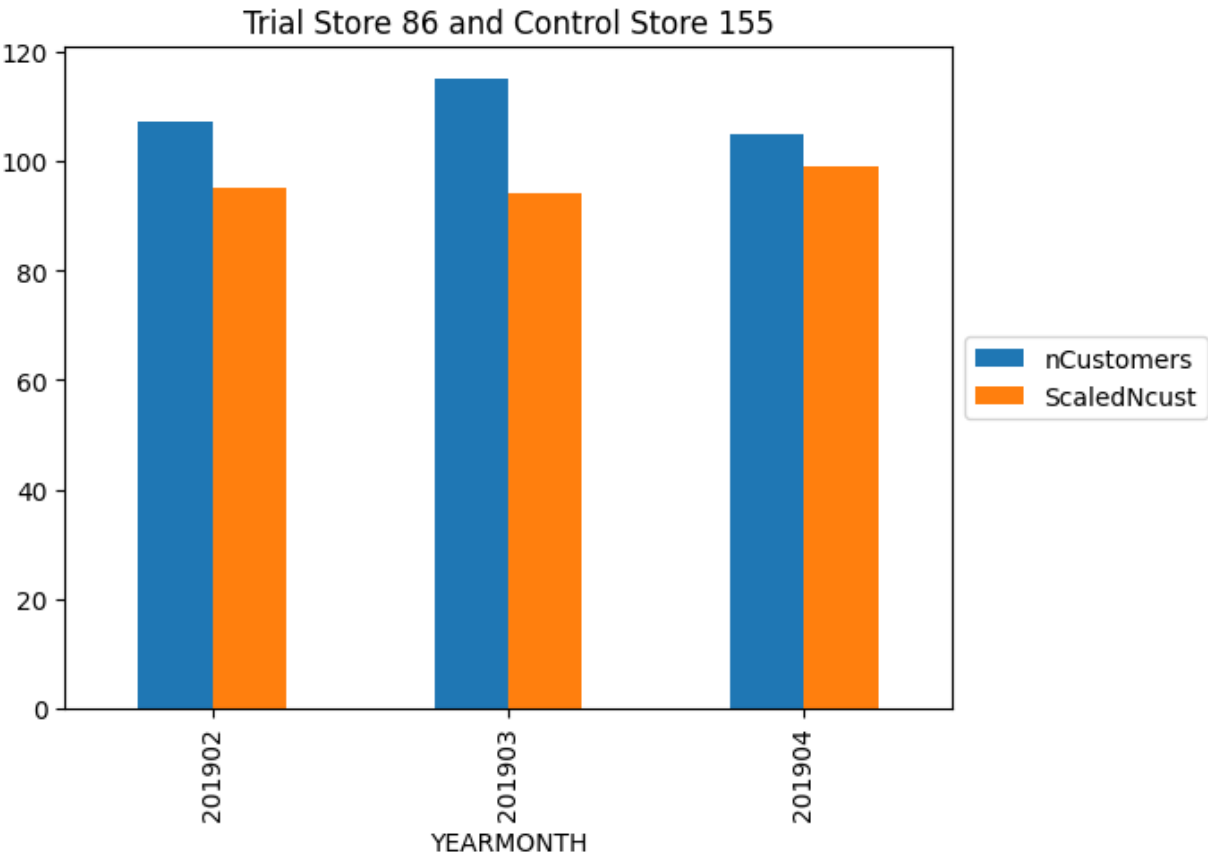
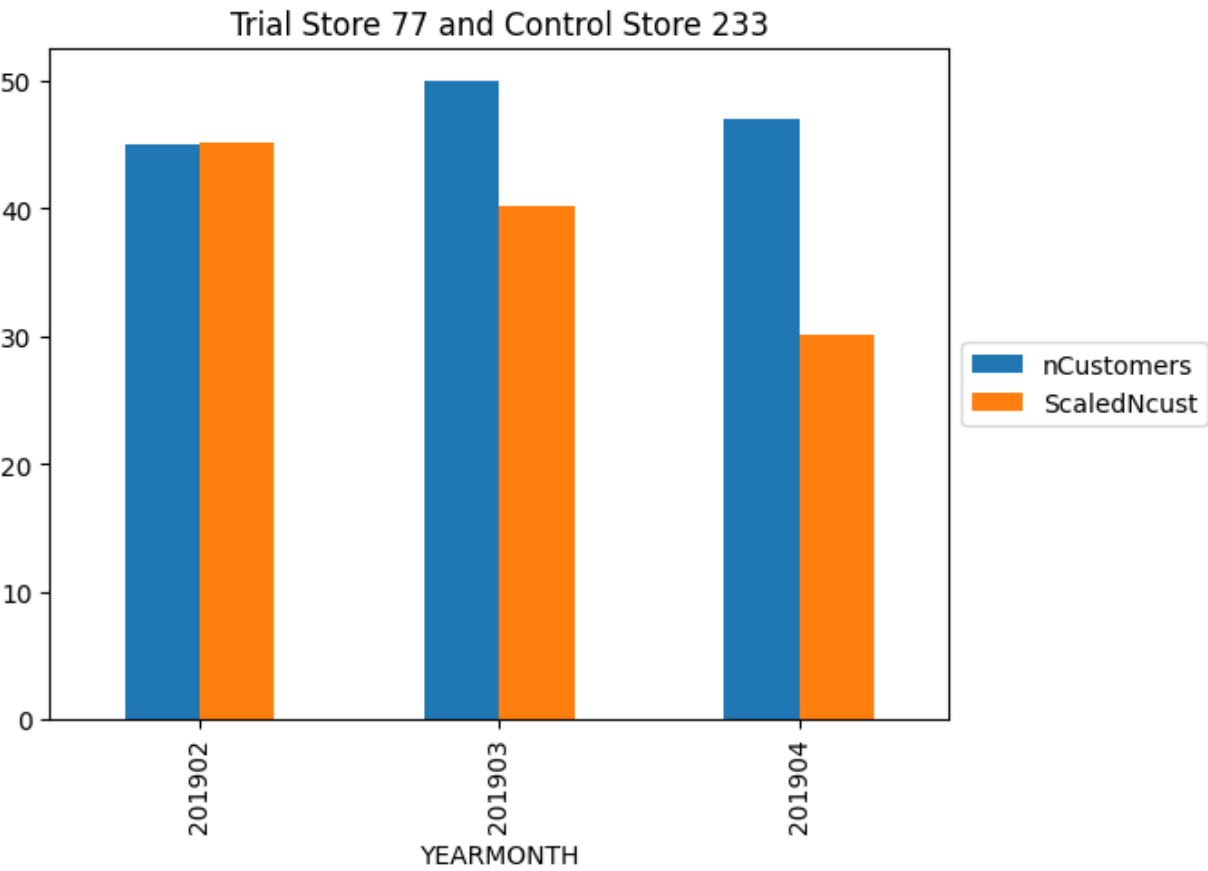
scaled_ncust_control_stores["ScaledNcust"] = scaled_ncust_control_stores.apply(scaler_c, axis=1)

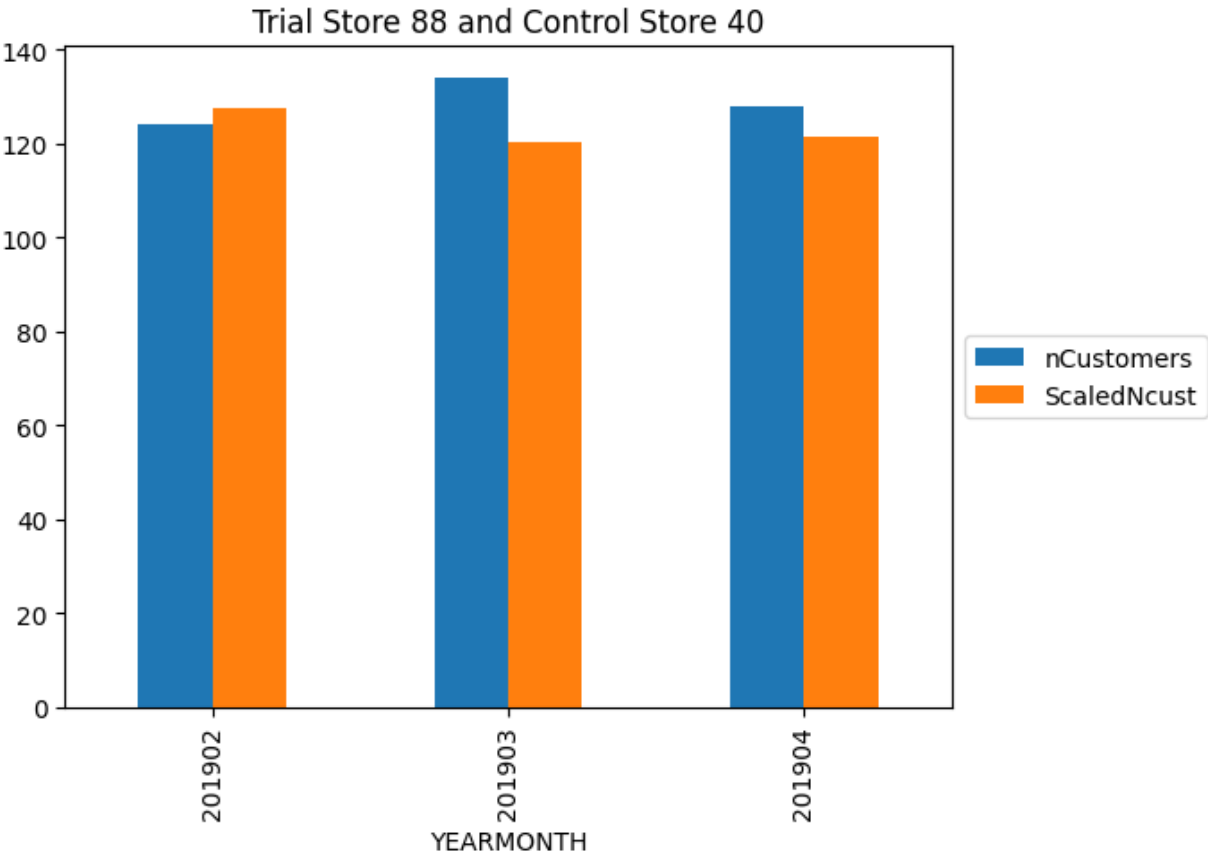
trial_scaled_ncust_control_stores = scaled_ncust_control_stores[(scaled_ncust_control_stores["STORE_NBR"]
pretrial_scaled_ncust_control_stores = scaled_ncust_control_stores[scaled_ncust_control_stores["STORE_NBR"]
    .isin([233, 155, 40])][
    "nCustomers"]

ncust_percentage_diff = {}

for trial, control in trial_control_dic.items():
    a = trial_scaled_ncust_control_stores[trial_scaled_ncust_control_stores["STORE_NBR"] == trial][
    "nCustomers"]
    b = trial_full_observ[trial_full_observ["STORE_NBR"] == trial][
    "nCustomers"]
    ncust_percentage_diff[trial] = b["nCustomers"].sum() / a["ScaledNcust"].sum()
    b[["YEARMONTH", "nCustomers"]].merge(a[["YEARMONTH", "ScaledNcust"]], on="YEARMONTH", how="left")
```

```
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
plt.title("Trial Store "+str(trial)+" and Control Store "+str(control))
```





```
In [42]: ncust_percentage_diff
```

Out[42]: {77: 1.2306529009742622, 86: 1.1354166666666667, 88: 1.0444876946258161}

```
In [43]: temp1 = scaled_ncust_control_stores.sort_values(by=["STORE_NBR", "YEARMONTH"],
temp2 = full_observ[full_observ["STORE_NBR"].isin([77,86,88])][["STORE_NBR", "t_nCustomers", "t_nCust_Percentage_Diff"]]
scaledncust_vs_trial = pd.concat([temp1, temp2], axis=1)
scaledncust_vs_trial.columns = ["c_STORE_NBR", "YEARMONTH", "c_ScaledNcust", "t_STORE_NBR", "t_nCustomers", "t_nCust_Percentage_Diff"]
scaledncust_vs_trial["nCust_Percentage_Diff"] = (scaledncust_vs_trial["t_nCust_Percentage_Diff"] - scaledncust_vs_trial["c_ScaledNcust"] / scaledncust_vs_trial["t_nCustomers"])

scaledncust_vs_trial["trial_period"] = scaledncust_vs_trial["YEARMONTH"].apply(lambda x: "201902" if x == "201902" else "201903" if x == "201903" else "201904")
scaledncust_vs_trial[scaledncust_vs_trial["trial_period"] == "trial"]
```

Out[43]:

	c_STORE_NBR	YEARMONTH	c_ScaledNcust	t_STORE_NBR	t_nCustomers	nCust_Percentage_Diff
7	233	201902	45.151007	77	45	-0.000000
8	233	201903	40.134228	77	50	-0.000000
9	233	201904	30.100671	77	47	0.000000
19	155	201902	95.000000	86	107	-0.000000
20	155	201903	94.000000	86	115	0.000000
21	155	201904	99.000000	86	105	0.000000
31	40	201902	127.610209	88	124	-0.000000
32	40	201903	120.464037	88	134	0.000000
33	40	201904	121.484919	88	128	0.000000

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.

```
In [44]: # Step 1
for num in [40, 155, 233]:
    print("Store", num)
    print(ttest_ind(pretrial_scaled_ncust_control_stores[pretrial_scaled_ncust_
        trial_scaled_ncust_control_stores[trial_scaled_ncust_control
        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_sto
        len(trial_scaled_ncust_control_stores[trial_scaled_ncust

Store 40
Ttest_indResult(statistic=0.644732693420032, pvalue=0.5376573016017127)

Store 155
Ttest_indResult(statistic=1.3888888888888882, pvalue=0.204345986327886)

Store 233
Ttest_indResult(statistic=0.8442563765225701, pvalue=0.4559280037660254)

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

```
In [45]: # Step 2
for trial, cont in trial_control_dic.items():
    print("Trial store:", trial, ", Control store:", cont)
    print(ttest_ind(pretrial_full_observ[pretrial_full_observ["STORE_NBR"] == t
        pretrial_scaled_ncust_control_stores[pretrial_scaled_ncust_c
        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(pretrial_full_observ[pretrial_full_obs
```


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=-7.648483953264653e-15, pvalue=0.999999999999994)

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

```
In [46]: # Step 3
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"] == trial]:
        pdif = scaledncust_vs_trial[(scaledncust_vs_trial["YEARMONTH"] == t_month)]
        print(t_month, ":", (float(pdif["nCust_Percentage_Diff"].mean()) - 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 : -0.19886295797440687
201903 : 8.009609025380932
201904 : 16.114474772873923

Trial store: 86 , Control store: 155
201902 : 6.220524882227514
201903 : 10.52599074274189
201904 : 3.0763575852842706

Trial store: 88 , Control store: 40
201902 : -0.3592881735131531
201903 : 1.2575196020616801
201904 : 0.6092905590514273

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

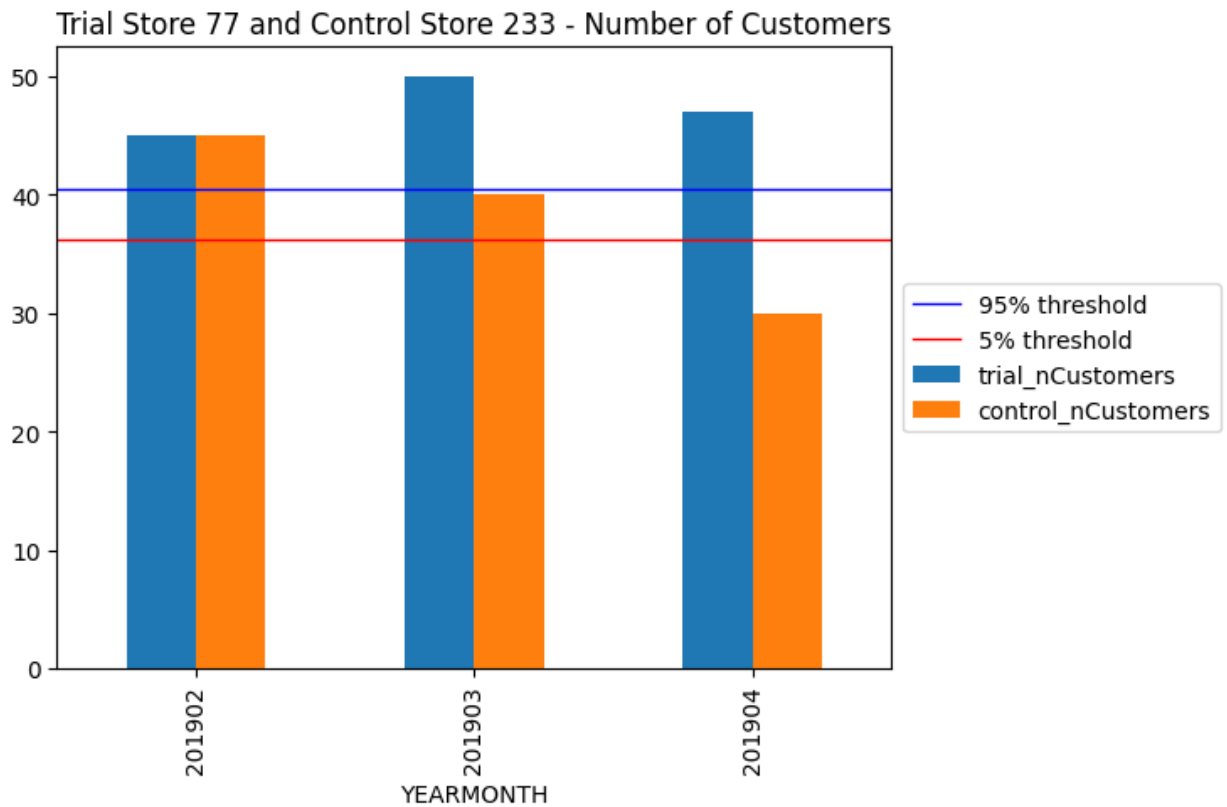
Feb, March and April trial months for trial store 86

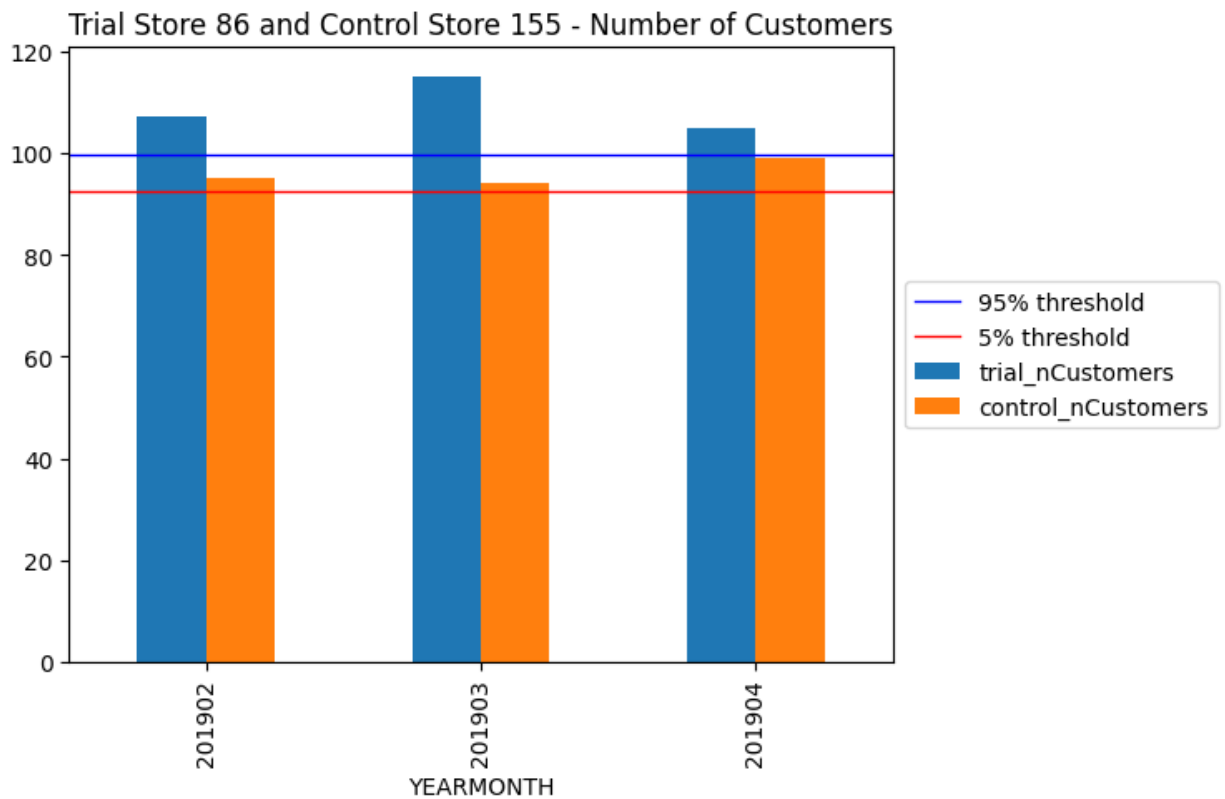
```
In [47]: for trial, control in trial_control_dic.items():
          a = trial_scaled_ncust_control_stores[trial_scaled_ncust_control_stores["STORE_NBR"] == trial]
          b = trial_full_observ[trial_full_observ["STORE_NBR"] == trial][["STORE_NBR", "YEARMONTH"]]
```

```

comb = b[["YEARMONTH", "trial_nCustomers"]].merge(a[["YEARMONTH", "control_nCustomers"]])
comb.plot.bar()
cont_sc_ncust = trial_scaled_ncust_control_stores[trial_scaled_ncust_control_stores["c_STORE_NBR"] == control]
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)+" - nCustomers")
plt.savefig("TS {} and CS {} - nCustomers.png".format(trial,control), bbox_

```





Insights :-

1. 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.

- ● ◆ ● ● ● ◆ ● ●