Task2

June 6, 2025

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

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

[3]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NATURAL	YOUNG SINGLES/COUPLES	Premium
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	RRD	YOUNG SINGLES/COUPLES	Mainstream
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	GRNWVES	YOUNG FAMILIES	Budget
3	1003	2019-03-08	1	4	106	Natural ChipCo Hony Soy Chckn175g	1	3.0	175	NATURAL	YOUNG FAMILIES	Budget
4	1004	2018-11-02	1	5	96	WW Original Stacked Chips 160g	1	1.9	160	WOOLWORTHS	OLDER SINGLES/COUPLES	Mainstream

Data Exploration

[4]:
```
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
```

[5]:
```
qvi.describe().T
```

[5]:

	count	mean	std	min	25%	50%	75%	max
LYLTY_CARD_NBR	264834.0	135548.793331	80579.898912	1000.0	70021.0	130357.0	203094.00	2373711.0
STORE_NBR	264834.0	135.079423	76.784063	1.0	70.0	130.0	203.00	272.0
TXN_ID	264834.0	135157.623236	78132.920436	1.0	67600.5	135136.5	202699.75	2415841.0
PROD_NBR	264834.0	56.583554	32.826444	1.0	28.0	56.0	85.00	114.0
PROD_QTY	264834.0	1.905813	0.343436	1.0	2.0	2.0	2.00	5.0
TOT_SALES	264834.0	7.299346	2.527241	1.5	5.4	7.4	9.20	29.5
PACK_SIZE	264834.0	182.425512	64.325148	70.0	150.0	170.0	175.00	380.0

Checking Missing Values

[6]:
```
qvi.isnull().sum()
```

[6]:
```
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
```

[10]:
```
qvi["DATE"] = pd.to_datetime(qvi["DATE"])
qvi["YEARMONTH"] = qvi["DATE"].dt.strftime("%Y%m").astype("int")
```

Trial Observation

[8]:
```
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
```

[11]:
```
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
```

[12]:
```
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)
```

[12]:

	STORE_NBR	YEARMONTH	TOT_SALES	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	1.061224	1.265306	3.337097
1	1	201808	176.1	42	1.023810	1.285714	3.261111
2	1	201809	278.8	59	1.050847	1.271186	3.717333
3	1	201810	188.1	44	1.022727	1.318182	3.243103
4	1	201811	192.6	46	1.021739	1.239130	3.378947
5	1	201812	189.6	42	1.119048	1.357143	3.326316
6	1	201901	154.8	35	1.028571	1.200000	3.685714
12	2	201807	150.8	39	1.051282	1.179487	3.278261

[13]:
```
def calcCorrTable(metricCol, storeComparison, inputTable=pretrial_full_observ):
  control_store_nbrs = inputTable[~inputTable["STORE_NBR"].isin([77, 86,␣
 ↪88])]["STORE_NBR"].unique()
  corrs = pd.DataFrame(columns = ["YEARMONTH", "Trial_Str", "Ctrl_Str",␣
 ↪"Corr_Score"])
  trial_store = inputTable[inputTable["STORE_NBR"] ==␣
 ↪storeComparison][metricCol].reset_index()
  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].
 ↪reset_index()
    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
```

[14]:
```
corr_table = pd.DataFrame()
for trial_num in [77, 86, 88]:
    corr_table = pd.concat([corr_table, calcCorrTable(["TOT_SALES",␣
 ↪"nCustomers", "nTxnPerCust", "nChipsPerTxn", "avgPricePerUnit"], trial_num)])
```

```
corr_table.head(8)
```

```
C:\Users\Harsh\AppData\Local\Temp\ipykernel_11136\3514888743.py:12:
FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
  corrs = pd.concat([corrs, concat_df])
C:\Users\Harsh\AppData\Local\Temp\ipykernel_11136\3514888743.py:12:
FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
  corrs = pd.concat([corrs, concat_df])
C:\Users\Harsh\AppData\Local\Temp\ipykernel_11136\3514888743.py:12:
FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
  corrs = pd.concat([corrs, concat_df])
```

[14]:

	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

[15]:

```
def calculateMagnitudeDistance(metricCol, storeComparison,␣
↪inputTable=pretrial_full_observ):
    control_store_nbrs = inputTable[~inputTable["STORE_NBR"].isin([77, 86,␣
↪88])]["STORE_NBR"].unique()
    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].
↪reset_index()[metricCol] - inputTable[inputTable["STORE_NBR"] == control].
↪reset_index()[metricCol])
        concat_df["YEARMONTH"] = list(inputTable[inputTable["STORE_NBR"] ==␣
↪storeComparison]["YEARMONTH"])
        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
```

[16]:
```
dist_table = pd.DataFrame()
for trial_num in [77, 86, 88]:
    dist_table = pd.concat([dist_table,␣
 ↪calculateMagnitudeDistance(["TOT_SALES", "nCustomers", "nTxnPerCust",␣
 ↪"nChipsPerTxn", "avgPricePerUnit"], trial_num)])

dist_table.head(8)
dist_table
```

[16]:

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

	YEARMONTH	Trial_Str	Ctrl_Str	magnitude
0	201807	77	1	0.899443
1	201808	77	1	0.915588
2	201809	77	1	0.844647
3	201810	77	1	0.888283
4	201811	77	1	0.870296
..	…	…	…	…
2	201809	88	272	0.532198
3	201810	88	272	0.614780
4	201811	88	272	0.600181
5	201812	88	272	0.554630
6	201901	88	272	0.602678

[5397 rows x 9 columns]

[17]:
```
def combine_corr_dist(metricCol, storeComparison,␣
 ↪inputTable=pretrial_full_observ):
    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
```

[18]:
```
compare_metrics_table1 = pd.DataFrame()
for trial_num in [77, 86, 88]:
    compare_metrics_table1 = pd.concat([compare_metrics_table1,␣
↪combine_corr_dist(["TOT_SALES"], trial_num)])
```

```
C:\Users\Harsh\AppData\Local\Temp\ipykernel_11136\3514888743.py:12:
FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
  corrs = pd.concat([corrs, concat_df])
C:\Users\Harsh\AppData\Local\Temp\ipykernel_11136\3514888743.py:12:
FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
  corrs = pd.concat([corrs, concat_df])
C:\Users\Harsh\AppData\Local\Temp\ipykernel_11136\3514888743.py:12:
FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
  corrs = pd.concat([corrs, concat_df])
```

[19]:
```
corr_weight = 0.5
dist_weight = 1 - corr_weight
```

[20]:
```
grouped_comparison_table1 = compare_metrics_table1.groupby(["Trial_Str",␣
↪"Ctrl_Str"]).mean().reset_index()
grouped_comparison_table1["CompScore"] = (corr_weight *␣
↪grouped_comparison_table1["Corr_Score"]) + (dist_weight *␣
↪grouped_comparison_table1["magnitude"])
for trial_num in compare_metrics_table1["Trial_Str"].unique():
    print(grouped_comparison_table1[grouped_comparison_table1["Trial_Str"] ==␣
↪trial_num].sort_values(ascending=False, by="CompScore").head(), '\n')
```

	Trial_Str	Ctrl_Str	YEARMONTH	Corr_Score	magnitude	CompScore
218	77	233	201822.571429	1.0	0.986477	0.993238
239	77	255	201822.571429	1.0	0.979479	0.989739
177	77	188	201822.571429	1.0	0.977663	0.988831
49	77	53	201822.571429	1.0	0.976678	0.988339
120	77	131	201822.571429	1.0	0.976267	0.988134

	Trial_Str	Ctrl_Str	YEARMONTH	Corr_Score	magnitude	CompScore

```
356          86        109  201822.571429          1.0    0.966783    0.983391
401          86        155  201822.571429          1.0    0.965876    0.982938
464          86        222  201822.571429          1.0    0.962280    0.981140
467          86        225  201822.571429          1.0    0.960512    0.980256
471          86        229  201822.571429          1.0    0.951704    0.975852

     Trial_Str  Ctrl_Str      YEARMONTH  Corr_Score  magnitude  CompScore
551          88         40  201822.571429          1.0    0.941165    0.970582
538          88         26  201822.571429          1.0    0.904377    0.952189
582          88         72  201822.571429          1.0    0.903800    0.951900
517          88          4  201822.571429          1.0    0.903466    0.951733
568          88         58  201822.571429          1.0    0.891678    0.945839
```

[21]:

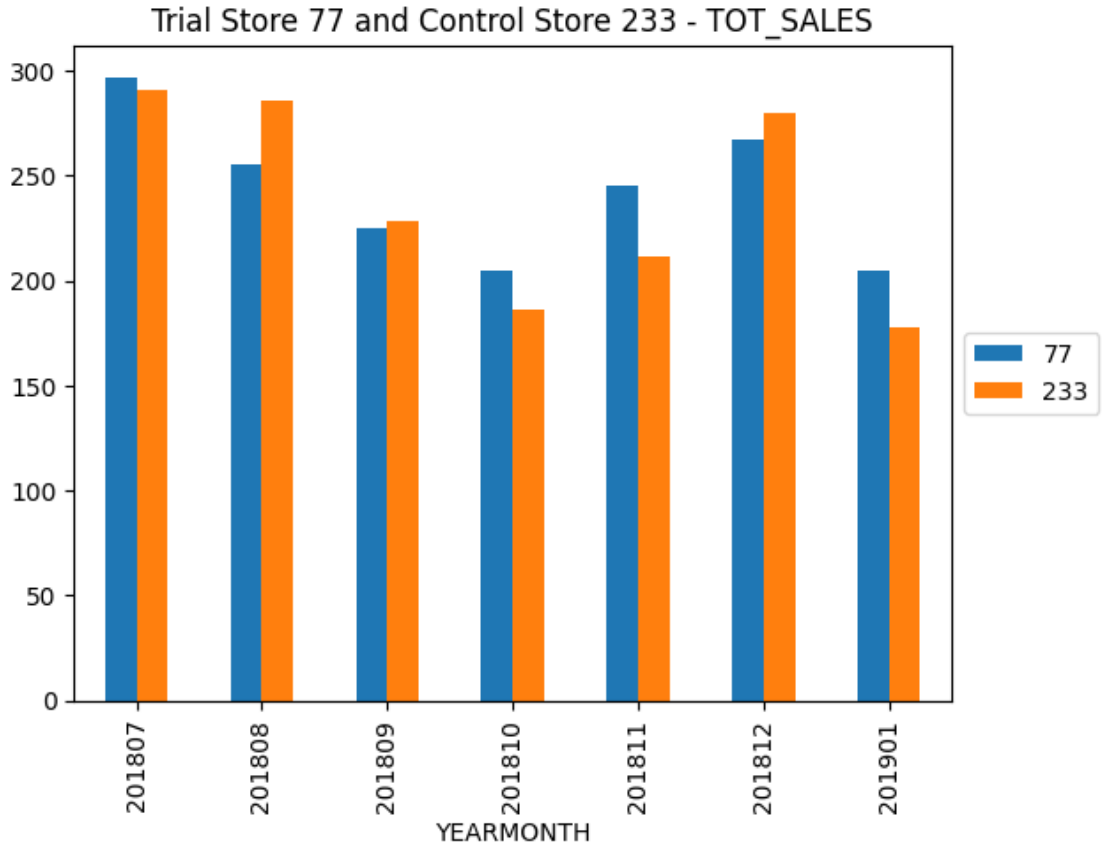
```python
compare_metrics_table2 = pd.DataFrame()
for trial_num in [77, 86, 88]:
    compare_metrics_table2 = pd.concat([compare_metrics_table2,␣
 ↪combine_corr_dist(["nCustomers"], trial_num)])
```

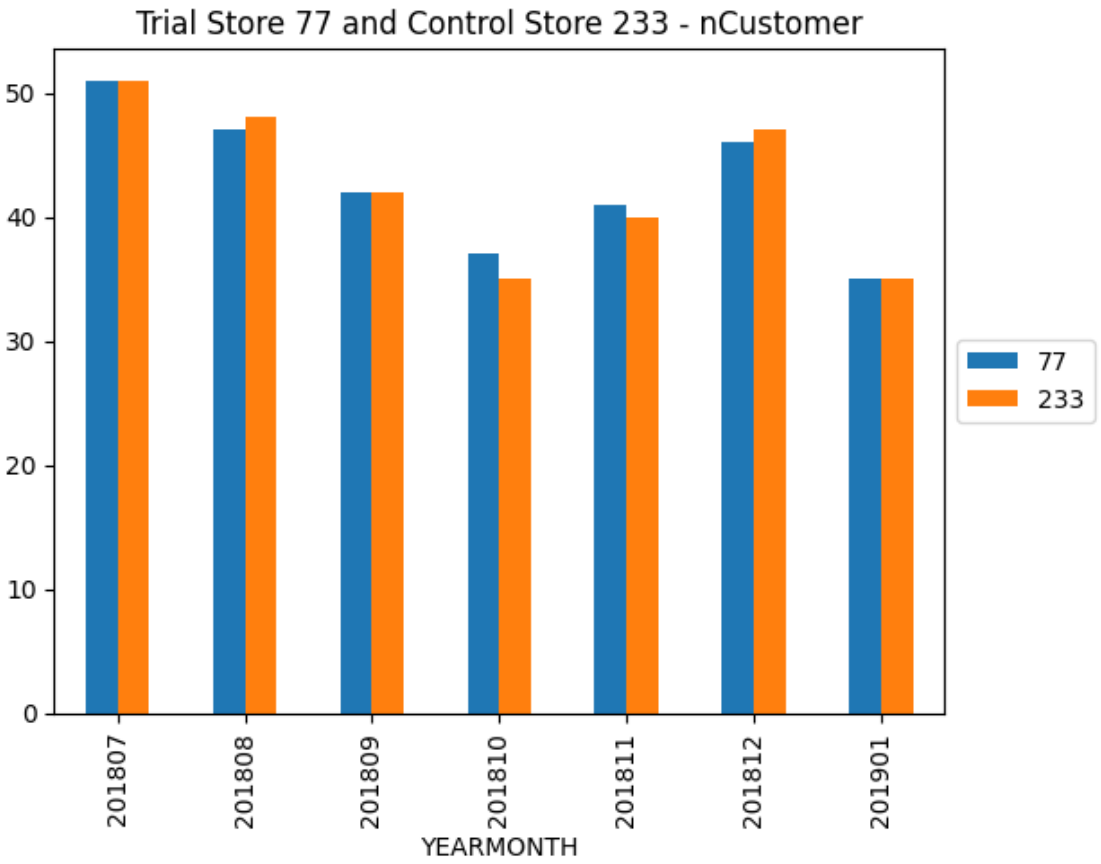
```
C:\Users\Harsh\AppData\Local\Temp\ipykernel_11136\3514888743.py:12:
FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
  corrs = pd.concat([corrs, concat_df])
C:\Users\Harsh\AppData\Local\Temp\ipykernel_11136\3514888743.py:12:
FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
  corrs = pd.concat([corrs, concat_df])
C:\Users\Harsh\AppData\Local\Temp\ipykernel_11136\3514888743.py:12:
FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
  corrs = pd.concat([corrs, concat_df])
```

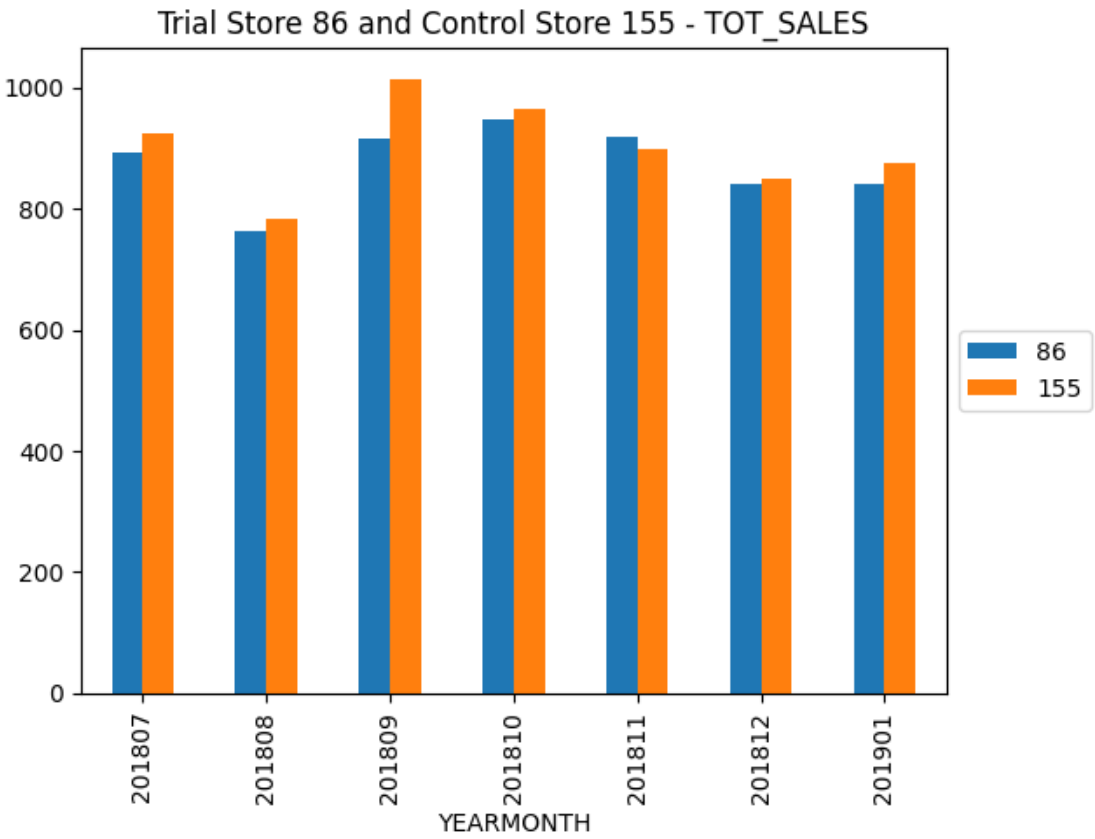
[22]:

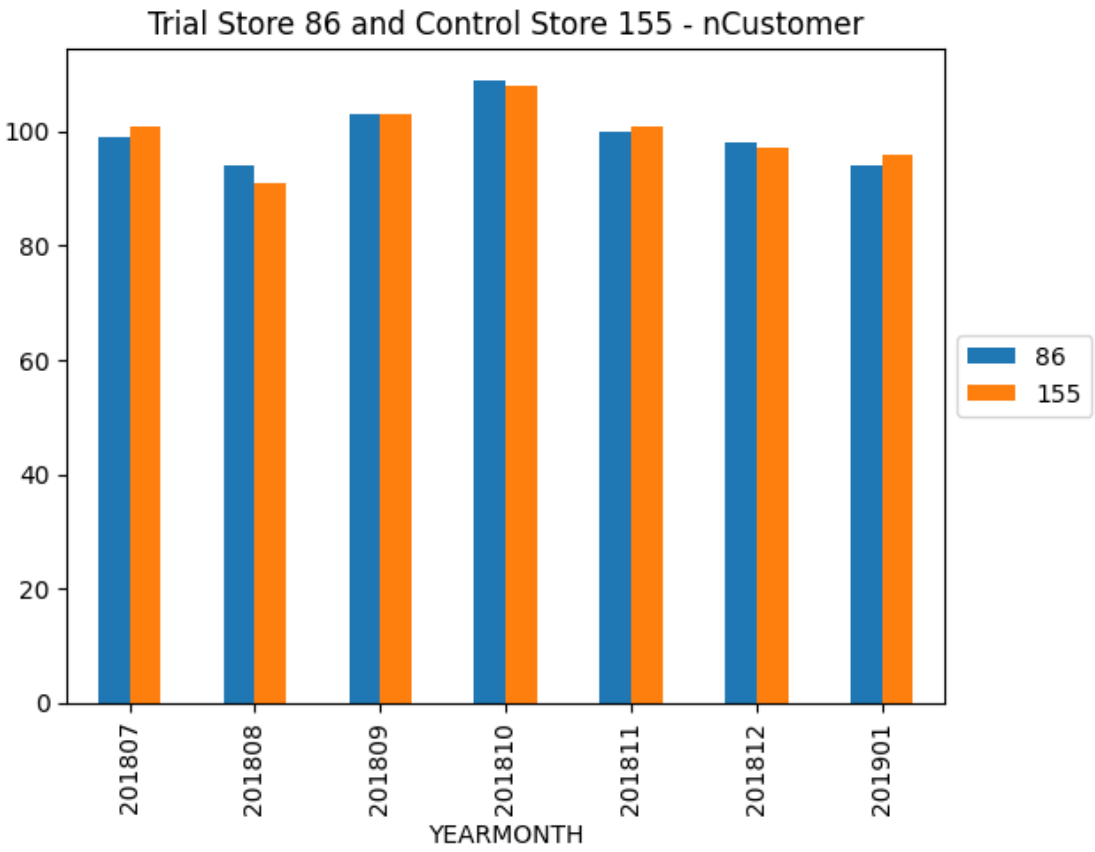
```python
trial_control_dic = {77:233, 86:155, 88:40}

for key, val in trial_control_dic.items():
    pretrial_full_observ[pretrial_full_observ["STORE_NBR"].isin([key, val])].
 ↪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()
```

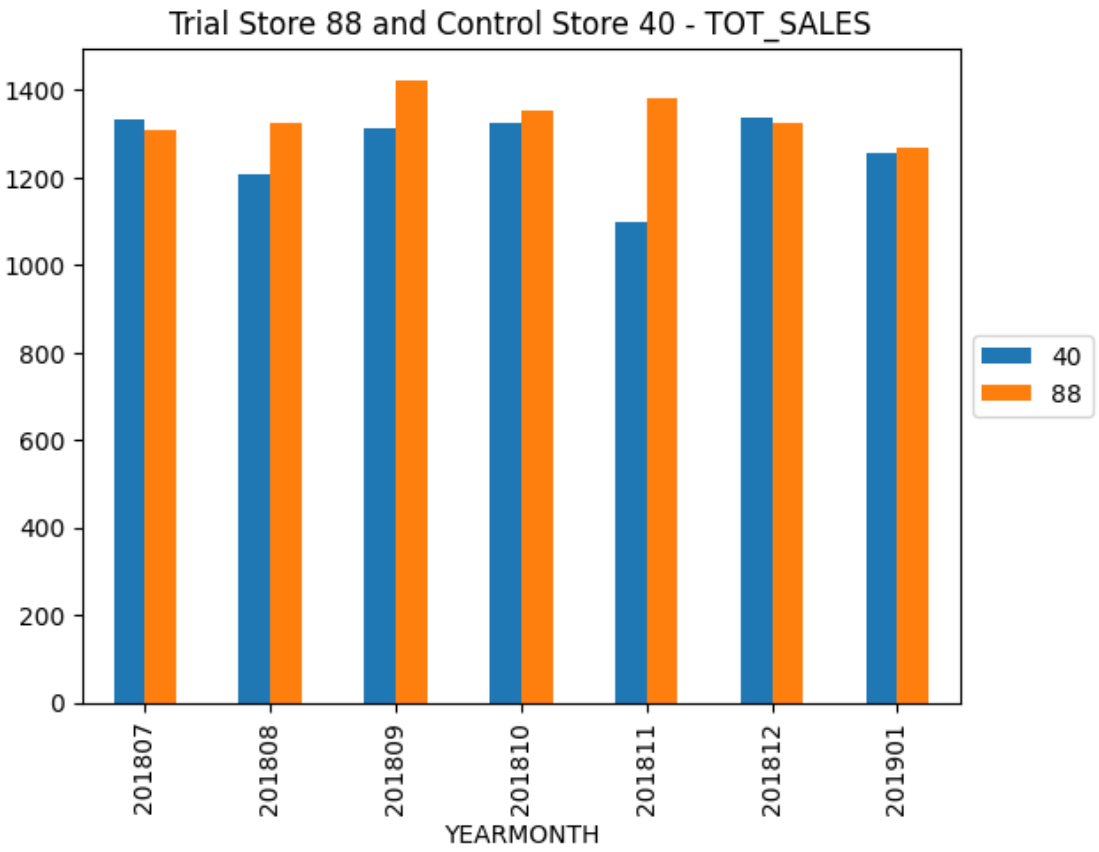
```
   pretrial_full_observ[pretrial_full_observ["STORE_NBR"].isin([key, val])].
↪groupby(
   ["YEARMONTH", "STORE_NBR"]).sum()["nCustomers"].unstack().plot.bar()
   plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
   plt.title("Trial Store "+str(key)+" and Control Store "+str(val)+" -␣
↪nCustomer")
   plt.show()
   print('\n')
```

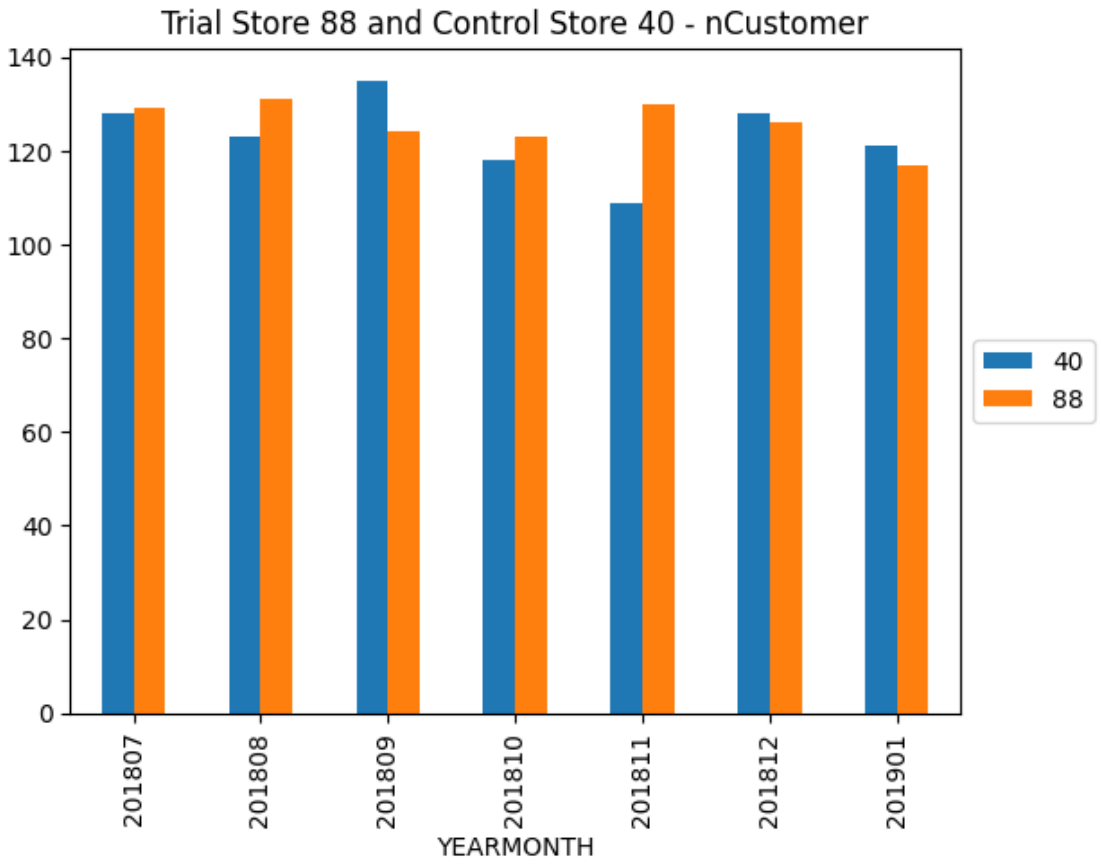



Trial Store 77 and Control Store 233 - nCustomer
50
40
30
20
10
0
201807
201808
201809
201810
201811
201812
201901
YEARMONTH
77
233


Trial Store 86 and Control Store 155 - TOT_SALES
1000
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

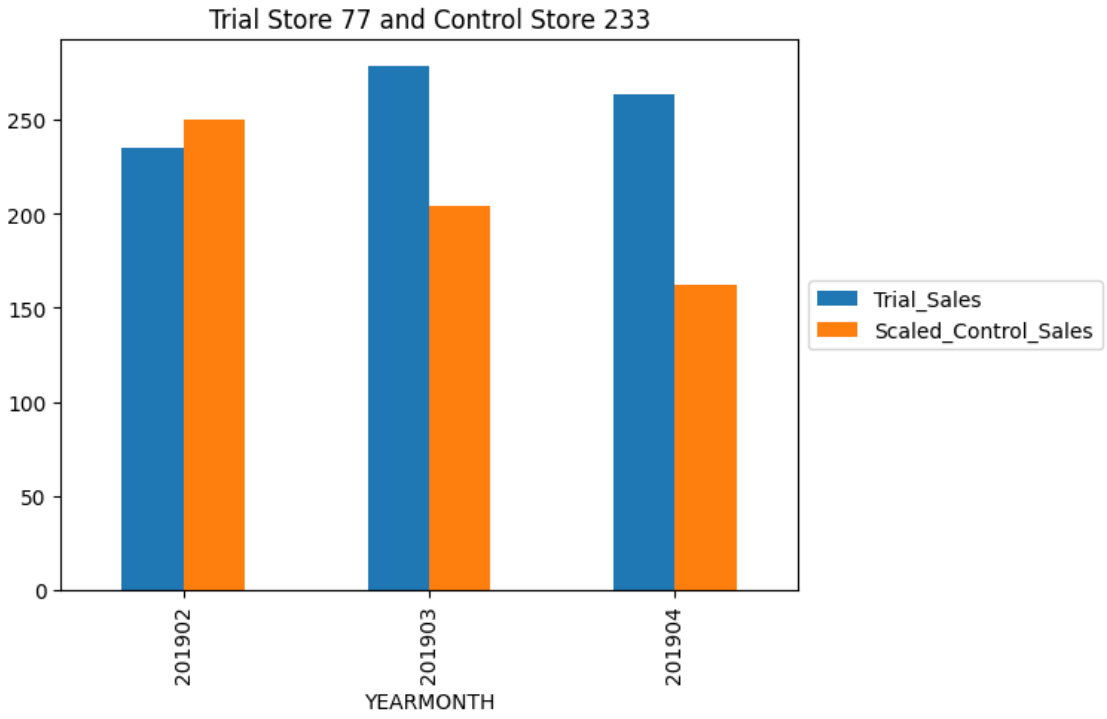

Trial Store 88 and Control Store 40 - TOT_SALES
1400
1200
1000
800
600
400
200
0
201807
201808
201809
201810
201811
201812
201901
YEARMONTH
40
88

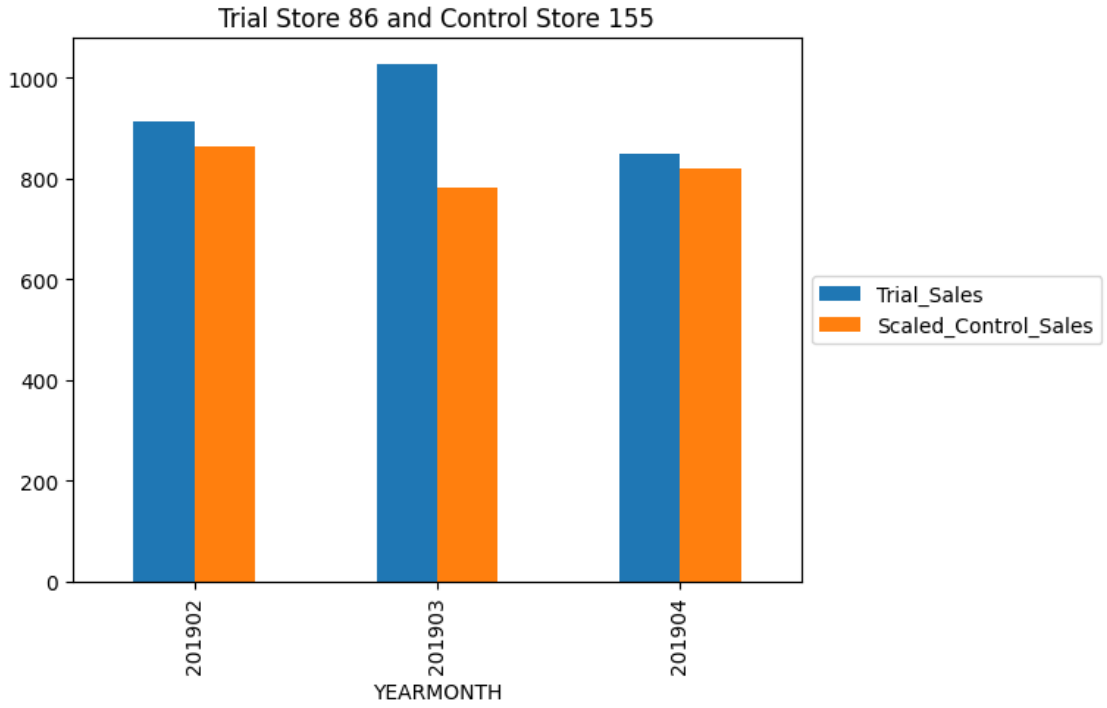


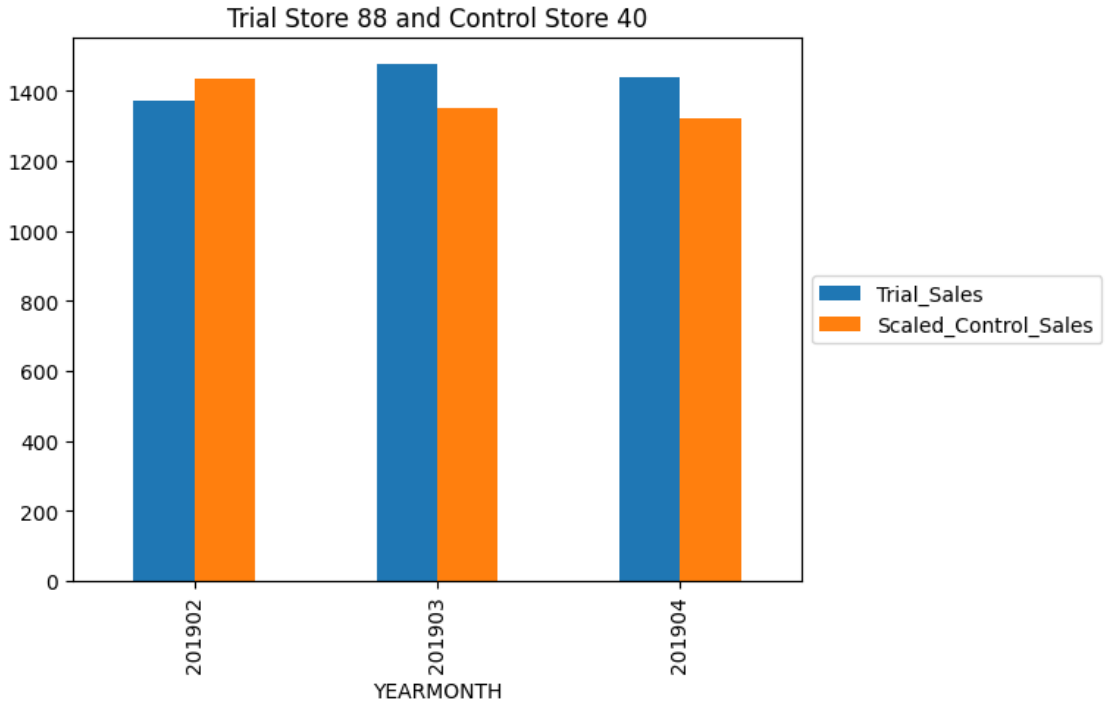
[26]:
```
sales_ratio_77 = pretrial_full_observ[pretrial_full_observ["STORE_NBR"] ==␣
↪77]["TOT_SALES"].sum() /␣
↪pretrial_full_observ[pretrial_full_observ["STORE_NBR"] == 233]["TOT_SALES"].
↪sum()
sales_ratio_86 = pretrial_full_observ[pretrial_full_observ["STORE_NBR"] ==␣
↪86]["TOT_SALES"].sum() /␣
↪pretrial_full_observ[pretrial_full_observ["STORE_NBR"] == 155]["TOT_SALES"].
↪sum()
sales_ratio_88 = pretrial_full_observ[pretrial_full_observ["STORE_NBR"] ==␣
↪88]["TOT_SALES"].sum() /␣
↪pretrial_full_observ[pretrial_full_observ["STORE_NBR"] == 40]["TOT_SALES"].
↪sum()
```

[27]:
```
trial_full_observ = full_observ[(full_observ["YEARMONTH"] >= 201902) &␣
↪(full_observ["YEARMONTH"] <= 201904)]
```

```
scaled_sales_control_stores = full_observ[full_observ["STORE_NBR"].isin([233,␣
↪155, 40])][["STORE_NBR", "YEARMONTH", "TOT_SALES"]]

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(lambda row: scaler(row), axis=1)

trial_scaled_sales_control_stores =␣
↪scaled_sales_control_stores[(scaled_sales_control_stores["YEARMONTH"] >=␣
↪201902) & (scaled_sales_control_stores["YEARMONTH"] <= 201904)]
pretrial_scaled_sales_control_stores =␣
↪scaled_sales_control_stores[scaled_sales_control_stores["YEARMONTH"] <␣
↪201902]

percentage_diff = {}

for trial, control in trial_control_dic.items():
    a =␣
↪trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["STORE_NBR"]␣
↪== control]
    b = trial_full_observ[trial_full_observ["STORE_NBR"] ==␣
↪trial][["STORE_NBR", "YEARMONTH", "TOT_SALES"]]
    percentage_diff[trial] = b["TOT_SALES"].sum() / a["ScaledSales"].sum()
    b[["YEARMONTH", "TOT_SALES"]].merge(a[["YEARMONTH",␣
↪"ScaledSales"]],on="YEARMONTH").set_index("YEARMONTH").
↪rename(columns={"ScaledSales":"Scaled_Control_Sales", "TOT_SALES":
↪"Trial_Sales"}).plot.bar()
    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
250
200
150
100
50
0
201902
201903
201904
YEARMONTH
Trial_Sales
Scaled_Control_Sales


Trial Store 86 and Control Store 155
1000
800
600
400
200
0
201902
201903
201904
YEARMONTH
Trial_Sales
Scaled_Control_Sales



[28]:
```
temp1 = scaled_sales_control_stores.sort_values(by=["STORE_NBR", "YEARMONTH"],␣
 ↪ascending=[False, True]).reset_index().drop(["TOT_SALES", "index"], axis=1)
temp2 = full_observ[full_observ["STORE_NBR"].isin([77,86,88])][["STORE_NBR",␣
 ↪"YEARMONTH", "TOT_SALES"]].reset_index().drop(["index", "YEARMONTH"], axis=1)
scaledsales_vs_trial = pd.concat([temp1, temp2], axis=1)
scaledsales_vs_trial.columns = ["c_STORE_NBR", "YEARMONTH", "c_ScaledSales",␣
 ↪"t_STORE_NBR", "t_TOT_SALES"]
scaledsales_vs_trial["Sales_Percentage_Diff"] =␣
 ↪(scaledsales_vs_trial["t_TOT_SALES"] -␣
 ↪scaledsales_vs_trial["c_ScaledSales"]) /␣
 ↪(((scaledsales_vs_trial["t_TOT_SALES"] +␣
 ↪scaledsales_vs_trial["c_ScaledSales"])/2))
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 cell: label_period(cell))
scaledsales_vs_trial[scaledsales_vs_trial["trial_period"] == "trial"]
```

[28]:

	c_STORE_NBR	YEARMONTH	c_ScaledSales	t_STORE_NBR	t_TOT_SALES	Sales_Percentage_Diff	trial_period
7	233	201902	249.762622	77	235.0	-0.060907	trial
8	233	201903	203.802205	77	278.5	0.309755	trial
9	233	201904	162.345704	77	263.5	0.475075	trial
19	155	201902	864.522060	86	913.2	0.054764	trial
20	155	201903	780.320405	86	1026.8	0.272787	trial
21	155	201904	819.317024	86	848.2	0.034642	trial
31	40	201902	1434.399269	88	1370.2	-0.045781	trial
32	40	201903	1352.064709	88	1477.2	0.088458	trial
33	40	201904	1321.797762	88	1439.4	0.085182	trial

[29]:
```
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_control_stores["
  ↪== num]["ScaledSales"],
                   ␣
  ↪trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["STORE_NBR"]␣
  ↪== num]["ScaledSales"],
                    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_sales_control_stores[pretrial_scaled_sales_control_stores["STOR
  ↪== num]),
                        ␣
  ↪len(trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["STORE_NBR"]␣
  ↪== num])])-1))
```

```
Store 40
TtestResult(statistic=np.float64(-0.5958372343168558),
pvalue=np.float64(0.5722861621434027), df=np.float64(6.228548324256264))
```

```
Store 155
TtestResult(statistic=np.float64(1.4291956879290917),
pvalue=np.float64(0.1972705865160342), df=np.float64(6.794437403919926))

Store 233
TtestResult(statistic=np.float64(1.191102601097452),
pvalue=np.float64(0.2944500606486209), df=np.float64(4.355475642590669))

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

[30]:
```
a = pretrial_scaled_sales_control_stores[pretrial_scaled_sales_control_stores["STORE_NBR"] == 40]["ScaledSales"]
b = trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["STORE_NBR"] == 40]["ScaledSales"]
```

[31]:
```
# 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"] == trial]["TOT_SALES"],
                    pretrial_scaled_sales_control_stores[pretrial_scaled_sales_control_stores["STORE_NBR"] == cont]["ScaledSales"],
                    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_observ["STORE_NBR"] == trial])-1))
```

```
Trial store: 77 , Control store: 233
TtestResult(statistic=np.float64(-1.2533353315065932e-15),
pvalue=np.float64(0.999999999999999), df=np.float64(12.0))

Trial store: 86 , Control store: 155
TtestResult(statistic=np.float64(3.1048311203382156e-15),
pvalue=np.float64(0.9999999999999976), df=np.float64(12.0))

Trial store: 88 , Control store: 40
TtestResult(statistic=np.float64(-5.69358613974361e-15),
pvalue=np.float64(0.9999999999999956), df=np.float64(12.0))

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

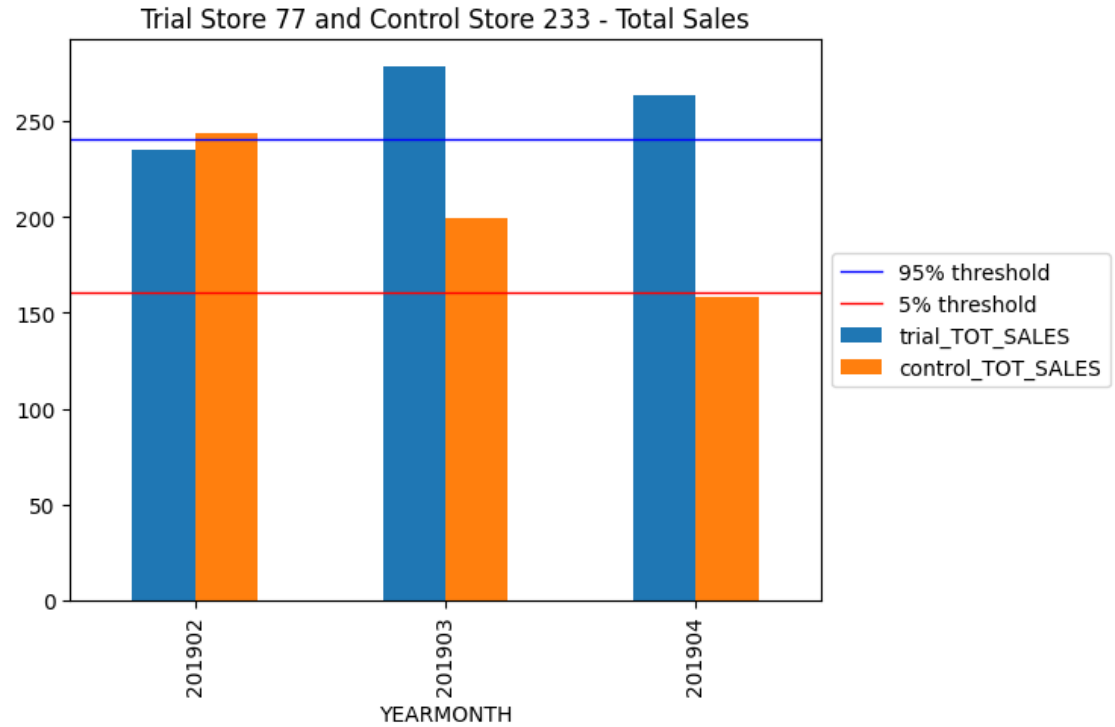
[32]:
```
# Step 3
for trial, cont in trial_control_dic.items():
    print("Trial store:", trial, ", Control store:", cont)
    temp_pre = scaledsales_vs_trial[(scaledsales_vs_trial["c_STORE_NBR"] ==␣
 ↪cont) & (scaledsales_vs_trial["trial_period"]=="pre")]
    std = temp_pre["Sales_Percentage_Diff"].std()
    mean = temp_pre["Sales_Percentage_Diff"].mean()
    for t_month in scaledsales_vs_trial[scaledsales_vs_trial["trial_period"] ==␣
 ↪"trial"]["YEARMONTH"].unique():
        pdif = scaledsales_vs_trial[(scaledsales_vs_trial["YEARMONTH"] ==␣
 ↪t_month) & (scaledsales_vs_trial["t_STORE_NBR"] ==␣
 ↪trial)]["Sales_Percentage_Diff"]
        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)
```

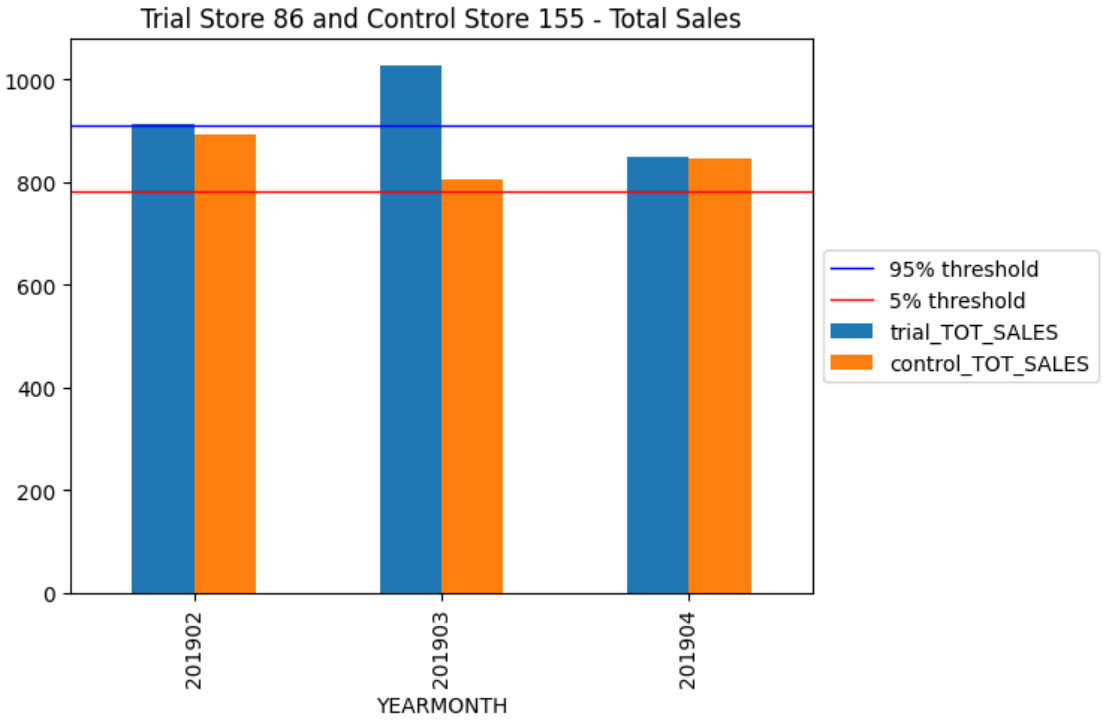
```
Trial store: 77 , Control store: 233
201902 : -0.7171038288055838
201903 : 3.035317928855674
201904 : 4.708944418758219


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:
1.9431802805153018
```

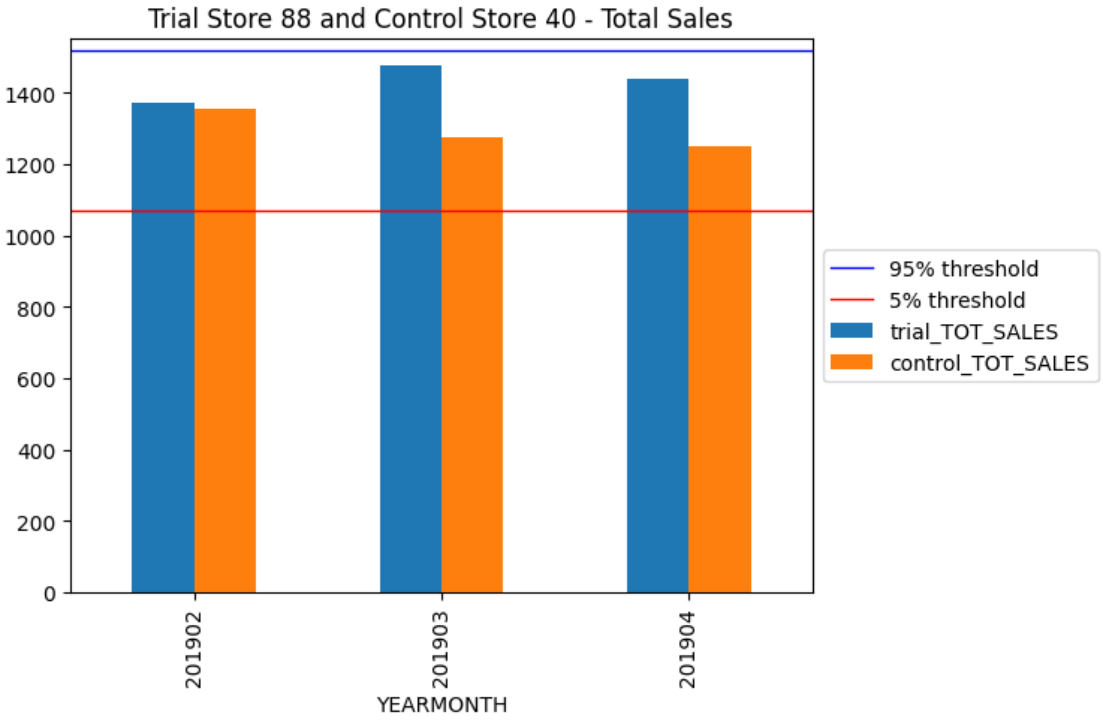
```
C:\Users\Harsh\AppData\Local\Temp\ipykernel_11136\3909951012.py:9:
FutureWarning: Calling float on a single element Series is deprecated and will
raise a TypeError in the future. Use float(ser.iloc[0]) instead
  print(t_month,":",(float(pdif)-mean)/std)
```

[33]:
```
for trial, control in trial_control_dic.items():
```

```
   a =␣
↪trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["STORE_NBR"]␣
↪== control].rename(columns={"TOT_SALES": "control_TOT_SALES"})
   b = trial_full_observ[trial_full_observ["STORE_NBR"] ==␣
↪trial][["STORE_NBR", "YEARMONTH", "TOT_SALES"]].rename(columns={"TOT_SALES":␣
↪"trial_TOT_SALES"})
   comb = b[["YEARMONTH", "trial_TOT_SALES"]].merge(a[["YEARMONTH",␣
↪"control_TOT_SALES"]],on="YEARMONTH").set_index("YEARMONTH")
   comb.plot.bar()
   cont_sc_sales =␣
↪trial_scaled_sales_control_stores[trial_scaled_sales_control_stores["STORE_NBR"]␣
↪== control]["TOT_SALES"]
   std = scaledsales_vs_trial[(scaledsales_vs_trial["c_STORE_NBR"] == control)␣
↪& (scaledsales_vs_trial["trial_period"]=="pre")]["Sales_Percentage_Diff"].
↪std()
   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)+" -␣
↪Total Sales")
   plt.savefig("TS {} and CS {} - TOT_SALES.png".format(trial,control),␣
↪bbox_inches="tight")
```




Trial Store 86 and Control Store 155 - Total Sales
1000
800
600
400
200
0
201902
201903
201904
YEARMONTH
95% threshold
5% threshold
trial_TOT_SALES
control_TOT_SALES


Trial Store 88 and Control Store 40 - Total Sales
1400
1200
1000
800
600
400
200
0
201902
201903
201904
YEARMONTH
95% threshold
5% threshold
trial_TOT_SALES
control_TOT_SALES

[ ]: