Experimentation and uplift testing

Select control stores 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

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

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
%matplotlib inline
```

In [62]:
```
df = pd.read_csv(r"C:\Users\shubh\Downloads\QVI_data.csv")
```

In [63]:
```
df
```

Out[63]:

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

264834 rows × 12 columns

In [64]:
```
df['DATE'] = pd.to_datetime(df['DATE'])
```

In [66]:
```
df['YEARMONTH'] = df['DATE'].dt.year*100 + df["DATE"].dt.month
```

In [67]:
```
df['YEARMONTH']
```

Out[67]:
```
0         201810
1         201809
2         201903
3         201903
4         201811
           ...
264829    201812
264830    201810
264831    201810
264832    201810
264833    201812
Name: YEARMONTH, Length: 264834, dtype: int32
```

In [68]:
```
# Calculate the measure per store per month
measureOverTime = (
    df.groupby(["STORE_NBR", "YEARMONTH"])
    .agg(
        totSales = ("TOT_SALES", "sum"),
        nCustomers = ("LYLTY_CARD_NBR", "nunique"),
        nTransactions = ("TXN_ID", "nunique"),
        qty = ("PROD_QTY", "sum")
    )
    .reset_index()
)
```

In [69]:
```
measureOverTime
```

Out[69]:

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTransactions	qty
0	1	201807	206.9	49	52	62
1	1	201808	176.1	42	43	54
2	1	201809	278.8	59	62	75
3	1	201810	188.1	44	45	58
4	1	201811	192.6	46	47	57
...	...	...	...	...	...	...
3164	272	201902	395.5	45	48	91
3165	272	201903	442.3	50	53	101
3166	272	201904	445.1	54	55	105
3167	272	201905	314.6	34	40	71
3168	272	201906	312.1	34	37	70

3169 rows × 6 columns

In [70]:
```
# Aggregate monthly store metrics
measure_over_time = (
    df.groupby(["STORE_NBR", "YEARMONTH"])
      .agg(
```

```
        totSales=("TOT_SALES", "sum"),
        nCustomers=("LYLTY_CARD_NBR", "nunique"),
        nTxn=("TXN_ID", "nunique"),
        qty=("PROD_QTY", "sum")
    )
    .reset_index()
)

measure_over_time["nTxnPerCust"] = measure_over_time["nTxn"] / measure_over_time["n
measure_over_time["nChipsPerTxn"] = measure_over_time["qty"] / measure_over_time["n
measure_over_time["avgPricePerUnit"] = measure_over_time["totSales"] / measure_over
```

In [71]:
```
pre_trial = measureOverTime[measureOverTime['YEARMONTH'] < 201902]
```

In [72]:
```
month_count = pre_trial.groupby('STORE_NBR')['YEARMONTH'].nunique()
```

In [73]:
```
month_count
```

Out[73]:
```
STORE_NBR
1      7
2      7
3      7
4      7
5      7
      ..
268    7
269    7
270    7
271    7
272    7
Name: YEARMONTH, Length: 271, dtype: int64
```

In [76]:
```
storesWithFullObs = month_count[month_count == 7].index.tolist()
```

In [ ]:
```

```

In [75]:
```
df['DATE'].sort_values()
```

Out[75]:
```
101808   2018-07-01
118872   2018-07-01
221754   2018-07-01
147394   2018-07-01
29702    2018-07-01
            ...
87112    2019-06-30
194066   2019-06-30
194046   2019-06-30
242433   2019-06-30
243910   2019-06-30
Name: DATE, Length: 264834, dtype: datetime64[ns]
```

In [80]:
```
import pandas as pd

def calculate_correlation(df, metric_col, trial_store):
    """
```

```
    df          → measure_over_time (monthly metrics)
    metric_col  → 'totSales', 'nCustomers', 'nTxnPerCust'
    trial_store → 77 / 86 / 88
    """

    stores = df["STORE_NBR"].unique()
    results = []

    # Extract trial store metric values
    trial_df = df[df["STORE_NBR"] == trial_store][["YEARMONTH", metric_col]]

    for store in stores:
        control_df = df[df["STORE_NBR"] == store][["YEARMONTH", metric_col]]

        # Align months
        aligned = pd.merge(trial_df, control_df, on="YEARMONTH", suffixes=("_trial"

        # Calculate correlation
        if aligned.shape[0] > 1 and aligned[metric_col + "_control"].std() > 0:
            corr_val = aligned[metric_col + "_trial"].corr(aligned[metric_col + "_c
        else:
            corr_val = 0

        results.append([trial_store, store, corr_val])

    return pd.DataFrame(results, columns=["trial_store", "control_store", "corr"])
```

In [82]:
```
corr_sales_77 = calculate_correlation(pre_trial, "totSales", 77)
corr_cust_77 = calculate_correlation(pre_trial, "nCustomers", 77)
corr_txn_77  = calculate_correlation(pre_trial, "nTransactions", 77)
```

In [84]:
```
corr_sales_86 = calculate_correlation(pre_trial, "totSales", 86)
corr_cust_86= calculate_correlation(pre_trial, "nCustomers", 86)
corr_txn_86  = calculate_correlation(pre_trial, "nTransactions", 86)
```

In [98]:
```
corr_sales_88 = calculate_correlation(pre_trial, "totSales", 88)
corr_cust_88= calculate_correlation(pre_trial, "nCustomers", 88)
corr_txn_88  = calculate_correlation(pre_trial, "nTransactions", 88)
```

In [99]:
```
# Normalise and combine the correlations
def combine_correlations(c1, c2, c3):
    df = c1.merge(c2, on=["trial_store", "control_store"], suffixes=("_sales", "_cu
    df = df.merge(c3, on=["trial_store", "control_store"])
    df = df.rename(columns={"corr": "corr_txn"})

    df["combined_similarity"] = df[["corr_sales", "corr_cust", "corr_txn"]].mean(ax
    return df
```

In [100…
```
similarity_77 = combine_correlations(corr_sales_77, corr_cust_77, corr_txn_77)
```

In [101…
```
similarity_86 = combine_correlations(corr_sales_86, corr_cust_86, corr_txn_86)
```

In [102…
```
similarity_88 = combine_correlations(corr_sales_88, corr_cust_88, corr_txn_88)
```

In [103…
```
# Select the best Control Store
def get_best_control(similarity_df, trial_store):
    sims = similarity_df[similarity_df["trial_store"] == trial_store]
    sims = sims[sims["control_store"] != trial_store]
    return sims.sort_values("combined_similarity", ascending=False).head(1)
```

In [104…
```
best_control_77 = get_best_control(similarity_77, 77)
print(best_control_77)
```

```
     trial_store  control_store  corr_sales  corr_cust  corr_txn  \
231           77            233    0.903774   0.990358  0.958422

     combined_similarity
231             0.950851
```

In [105…
```
best_control_86 = get_best_control(similarity_86, 86)
print(best_control_86)
```

```
     trial_store  control_store  corr_sales  corr_cust  corr_txn  \
153           86            155    0.877882   0.942876  0.642294

     combined_similarity
153             0.821017
```

In [106…
```
best_control_88 = get_best_control(similarity_88, 88)
print(best_control_88)
```

```
    trial_store  control_store  corr_sales  corr_cust  corr_txn  \
13           88             14    0.698557   0.942976  0.836849

    combined_similarity
13             0.826128
```

In [97]:
```
pre_trial[pre_trial["STORE_NBR"] == 88]
```

Out[97]:

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTransactions	qty
1001	88	201807	1310.0	129	153	306
1002	88	201808	1323.8	131	158	303
1003	88	201809	1423.0	124	157	318
1004	88	201810	1352.4	123	155	316
1005	88	201811	1382.8	130	156	314
1006	88	201812	1325.2	126	148	298
1007	88	201901	1266.4	117	144	292

In [107…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind

def run_trial_analysis(measure_over_time, merged, trial_store, control_store):
```

```
"""
Full Quantium Task 2 Analysis Pipeline:
- Plots
- Significance test
- Uplift calculation
- Driver metrics
"""

print("=====================================================")
print(f" TRIAL STORE {trial_store}  |  CONTROL STORE {control_store}")
print("=====================================================\n")

# ------------------------------
# Filter store-level data
# ------------------------------
trial_df = measure_over_time[measure_over_time["STORE_NBR"] == trial_store].cop
control_df = measure_over_time[measure_over_time["STORE_NBR"] == control_store]

# ------------------------------
# Define periods
# ------------------------------
trial_start = 201903

# ------------------------------
# Scale the control store
# ------------------------------
trial_pre = trial_df[trial_df["YEARMONTH"] < trial_start]["totSales"].sum()
control_pre = control_df[control_df["YEARMONTH"] < trial_start]["totSales"].sum

scale_factor = trial_pre / control_pre
control_df["scaled_sales"] = control_df["totSales"] * scale_factor

# ------------------------------
# Step 6: Plot trial vs control
# ------------------------------
plt.figure(figsize=(12, 6))
plt.plot(trial_df["YEARMONTH"], trial_df["totSales"],
         label=f"Trial Store {trial_store}", linewidth=3)
plt.plot(control_df["YEARMONTH"], control_df["scaled_sales"],
         label=f"Scaled Control Store {control_store}", linewidth=3)
plt.axvline(trial_start, color="red", linestyle="--", label="Trial Start")
plt.title(f"Monthly Sales: Trial vs Control (Store {trial_store})")
plt.xlabel("YearMonth")
plt.ylabel("Total Sales")
plt.legend()
plt.grid()
plt.show()

# ------------------------------
# Step 7: t-test on trial period
# ------------------------------
trial_period = trial_df[trial_df["YEARMONTH"] >= trial_start]
control_period = control_df[control_df["YEARMONTH"] >= trial_start]

# Convert monthly to daily values
trial_daily = trial_period["totSales"] / 30
```

```
control_daily = control_period["scaled_sales"] / 30

t_stat, p_value = ttest_ind(
    trial_daily, control_daily,
    equal_var=False, alternative="greater"
)

print("===== Welch T-Test (Is Trial > Control?) =====")
print(f"T-statistic : {t_stat:.4f}")
print(f"P-value     : {p_value:.6f}")
if p_value < 0.05:
    print("✓ Statistically Significant Improvement")
else:
    print("✗ No Significant Improvement")
print()

# ------------------------------
# Step 8: Uplift %
# ------------------------------
uplift = (
    trial_period["totSales"].sum() -
    control_period["scaled_sales"].sum()
) / control_period["scaled_sales"].sum() * 100

print("===== UPLIFT DURING TRIAL =====")
print(f"Uplift %: {uplift:.2f}%\n")

# ------------------------------
# DRIVER METRICS (during trial)
# ------------------------------
merged["YEAR_MONTH_NUM"] = merged["DATE"].dt.year * 100 + merged["DATE"].dt.mon

merged_trial = merged[
    (merged["YEAR_MONTH_NUM"] >= trial_start)
]

# Trial store metrics
m_t = merged_trial[merged_trial["STORE_NBR"] == trial_store]
m_c = merged_trial[merged_trial["STORE_NBR"] == control_store]

metrics = {
    "Total Customers": [
        m_t["LYLTY_CARD_NBR"].nunique(),
        m_c["LYLTY_CARD_NBR"].nunique()
    ],
    "Transactions per Customer": [
        m_t["TXN_ID"].nunique() / m_t["LYLTY_CARD_NBR"].nunique(),
        m_c["TXN_ID"].nunique() / m_c["LYLTY_CARD_NBR"].nunique(),
    ],
    "Units per Transaction": [
        m_t["PROD_QTY"].sum() / m_t["TXN_ID"].nunique(),
        m_c["PROD_QTY"].sum() / m_c["TXN_ID"].nunique(),
    ],
    "Avg Price per Unit": [
        m_t["TOT_SALES"].sum() / m_t["PROD_QTY"].sum(),
        m_c["TOT_SALES"].sum() / m_c["PROD_QTY"].sum(),
```

```
        ]
    }

    driver_df = pd.DataFrame(metrics, index=["Trial", "Control"])

    return {
        "trial_store": trial_store,
        "control_store": control_store,
        "uplift": uplift,
        "t_stat": t_stat,
        "p_value": p_value,
        "driver_metrics": driver_df
    }
```

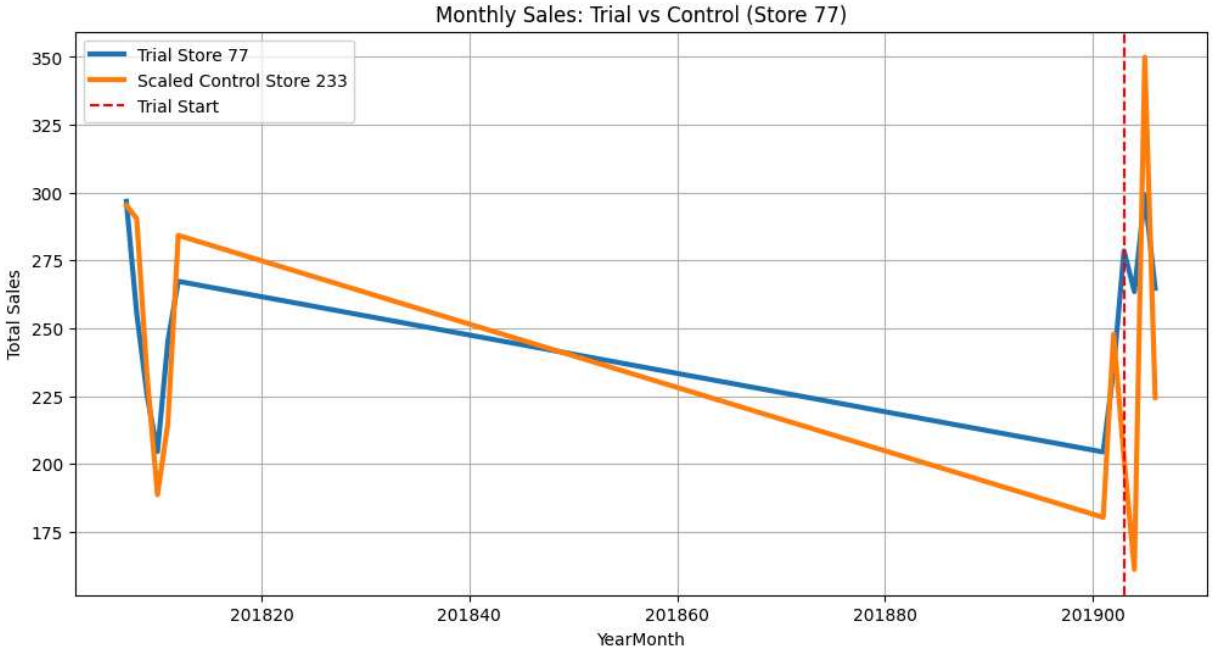
In [109…

```
merged = df.copy()
```

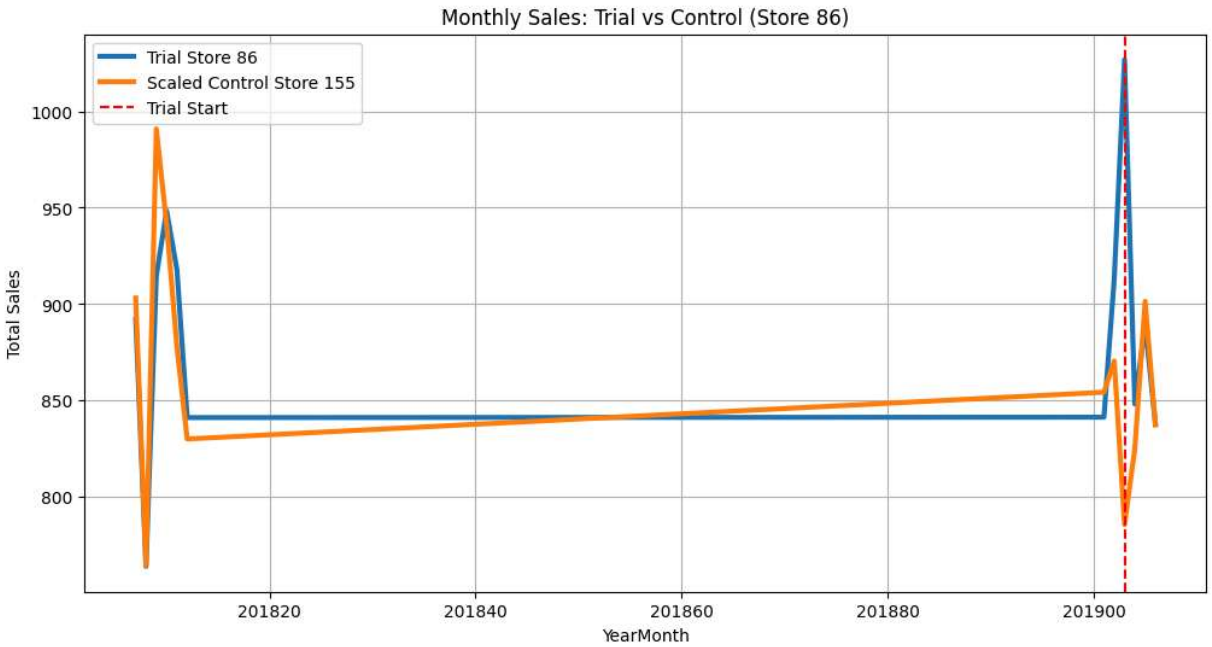
In [110…

```
controls = {
    77: 233,
    86: 155,
}

results = {}

for trial, control in controls.items():
    results[trial] = run_trial_analysis(
        measure_over_time, merged, trial, control
    )
```

```
==================================================
 TRIAL STORE 77  |  CONTROL STORE 233
==================================================
```



```
===== Welch T-Test (Is Trial > Control?) =====
T-statistic : 1.0136
P-value     : 0.190072
✗ No Significant Improvement

===== UPLIFT DURING TRIAL =====
Uplift %: 17.94%

======================================================
 TRIAL STORE 86  |  CONTROL STORE 155
======================================================
```




```
===== Welch T-Test (Is Trial > Control?) =====
T-statistic : 1.2712
P-value     : 0.131631
✗ No Significant Improvement

===== UPLIFT DURING TRIAL =====
Uplift %: 7.54%
```

In [111…

```
measure_over_time
```

Out[111…

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTxn	qty	nTxnPerCust	nChipsPe
0	1	201807	206.9	49	52	62	1.061224	1.19
1	1	201808	176.1	42	43	54	1.023810	1.25
2	1	201809	278.8	59	62	75	1.050847	1.20
3	1	201810	188.1	44	45	58	1.022727	1.28
4	1	201811	192.6	46	47	57	1.021739	1.21
…	…	…	…	…	…	…	…	
3164	272	201902	395.5	45	48	91	1.066667	1.89
3165	272	201903	442.3	50	53	101	1.060000	1.90
3166	272	201904	445.1	54	55	105	1.018519	1.90
3167	272	201905	314.6	34	40	71	1.176471	1.77
3168	272	201906	312.1	34	37	70	1.088235	1.89

3169 rows × 9 columns

In [112…

```
pre_trial = measure_over_time[measure_over_time["YEARMONTH"] < 201902]
```

In [119…

```
trial_stores = [77,86]
control_stores  = {
    77: 233,
    86: 155
}
```

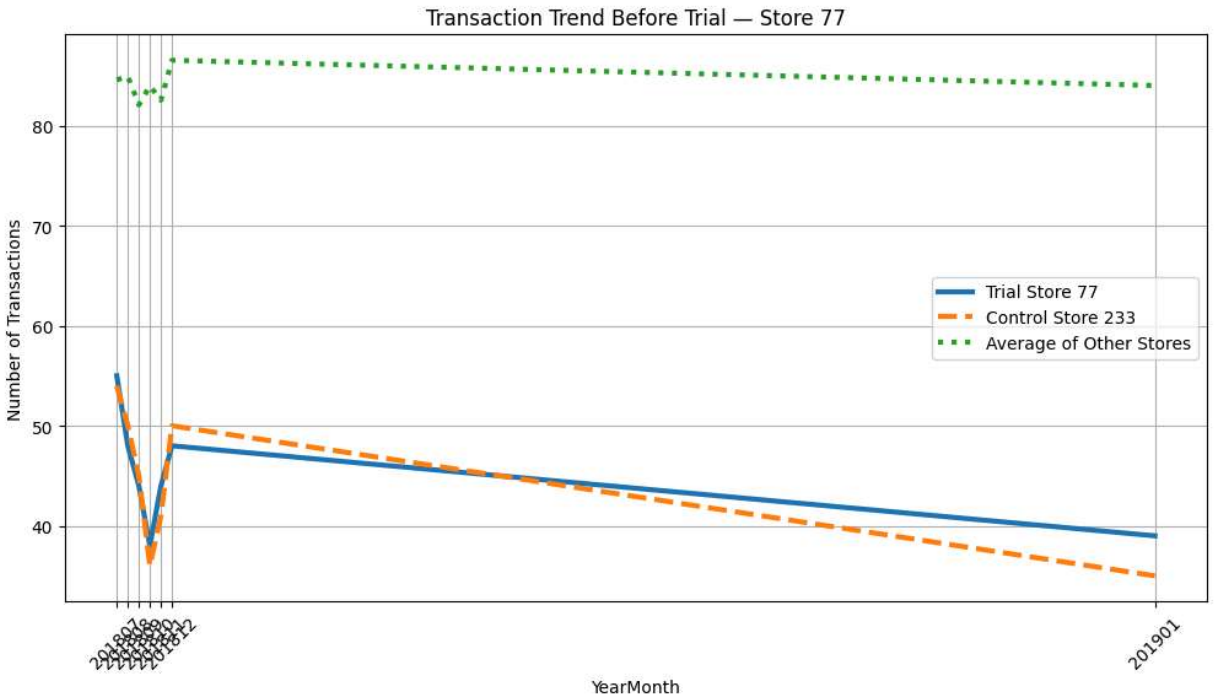
In [120…

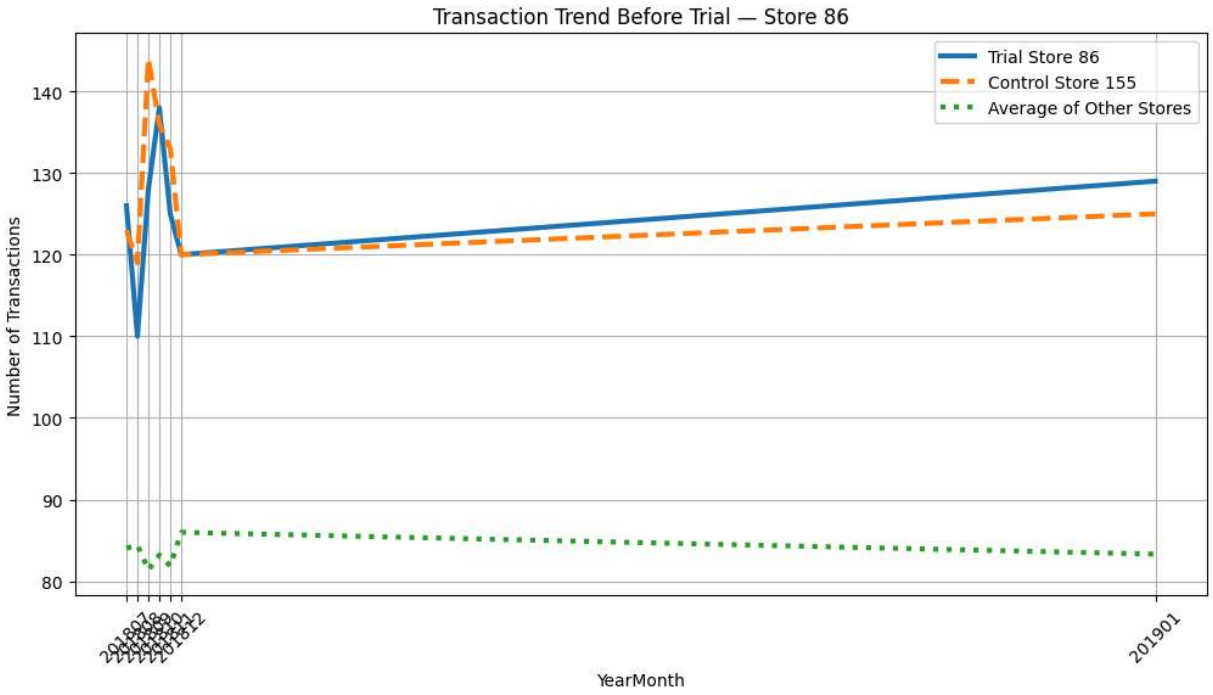
```
import matplotlib.pyplot as plt
import pandas as pd

def plot_transactions_trend(df, trial_store, control_store):
    # Filter data for required stores
    trial_df = df[df["STORE_NBR"] == trial_store]
    control_df = df[df["STORE_NBR"] == control_store]

    # All other stores for comparison
    other_df = df[(df["STORE_NBR"] != trial_store) &
                  (df["STORE_NBR"] != control_store)]

    # Group others as average
    others_grouped = (
        other_df.groupby("YEARMONTH")["nTxn"]
        .mean()
        .reset_index(name="avg_other_stores")
    )

    # Merge all three
    merged = (
```

```
        trial_df[["YEARMONTH", "nTxn"]]
        .rename(columns={"nTxn": "trial"})
        .merge(control_df[["YEARMONTH", "nTxn"]]
               .rename(columns={"nTxn": "control"}),
               on="YEARMONTH",
               how="left")
        .merge(others_grouped, on="YEARMONTH", how="left")
    )

    # Plot
    plt.figure(figsize=(12, 6))
    plt.plot(merged["YEARMONTH"], merged["trial"], label=f"Trial Store {trial_store
    plt.plot(merged["YEARMONTH"], merged["control"], label=f"Control Store {control
    plt.plot(merged["YEARMONTH"], merged["avg_other_stores"], label="Average of Oth

    plt.title(f"Transaction Trend Before Trial — Store {trial_store}")
    plt.xlabel("YearMonth")
    plt.ylabel("Number of Transactions")
    plt.xticks(merged["YEARMONTH"], rotation=45)
    plt.legend()
    plt.grid(True)
    plt.show()
```

In [121…

```
for trial in trial_stores:
    control = control_stores[trial]
    plot_transactions_trend(pre_trial, trial, control)
```



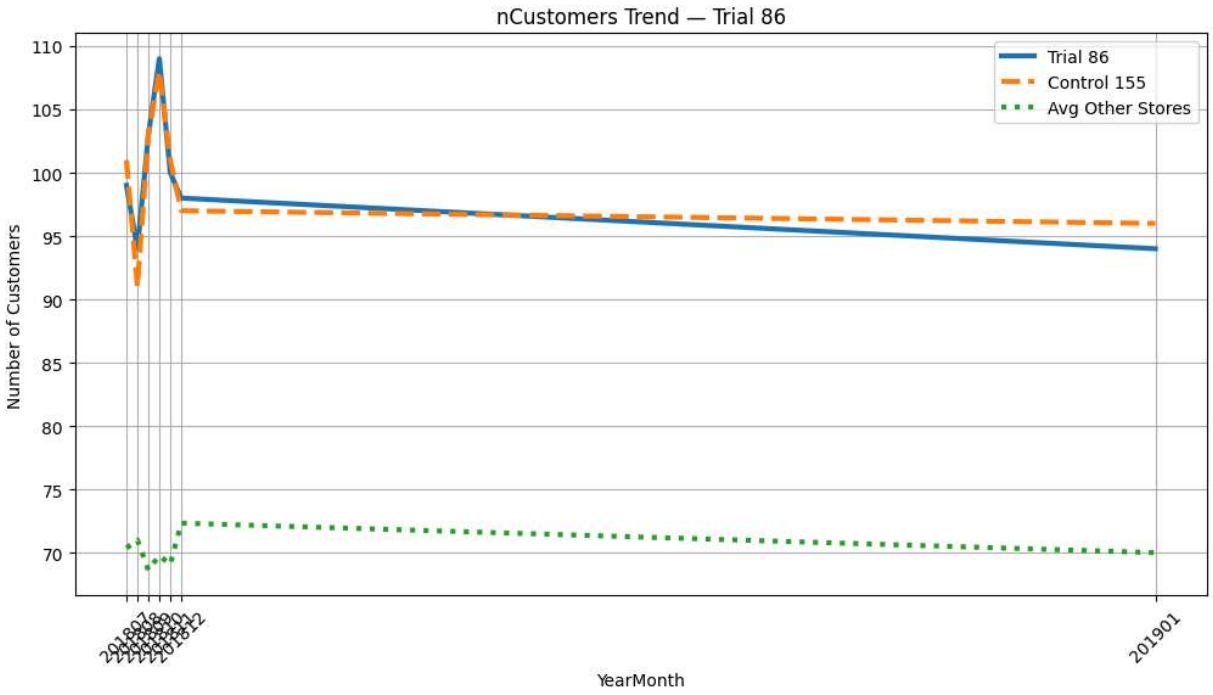


In [122…

```
def plot_customers(df, trial_store, control_store):
    # Trial store data
    trial = df[df["STORE_NBR"] == trial_store][["YEARMONTH", "nCustomers"]]

    # Control store data
    control = df[df["STORE_NBR"] == control_store][["YEARMONTH", "nCustomers"]]

    # Average of all other stores
    others = df[(df["STORE_NBR"] != trial_store) &
                (df["STORE_NBR"] != control_store)]
    others_avg = others.groupby("YEARMONTH")["nCustomers"].mean().reset_index()

    # Plot
    plt.figure(figsize=(12, 6))
    plt.plot(trial["YEARMONTH"], trial["nCustomers"], label=f"Trial {trial_store}",
    plt.plot(control["YEARMONTH"], control["nCustomers"], label=f"Control {control_
    plt.plot(others_avg["YEARMONTH"], others_avg["nCustomers"], label="Avg Other St

    plt.title(f"nCustomers Trend – Trial {trial_store}")
    plt.xlabel("YearMonth")
    plt.ylabel("Number of Customers")
    plt.xticks(trial["YEARMONTH"], rotation=45)
    plt.grid(True)
    plt.legend()
    plt.show()
```

In [123…

```
plot_customers(pre_trial, 77, 233)
plot_customers(pre_trial, 86, 155)
```



trial store and control store followed similar customer trends and transaction trend before the trial

In [128…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def run_sales_uplift(measure_df, trial_store, control_store):

    # Pre-trial data
    pre = measure_df[measure_df["YEARMONTH"] < 201902]
```

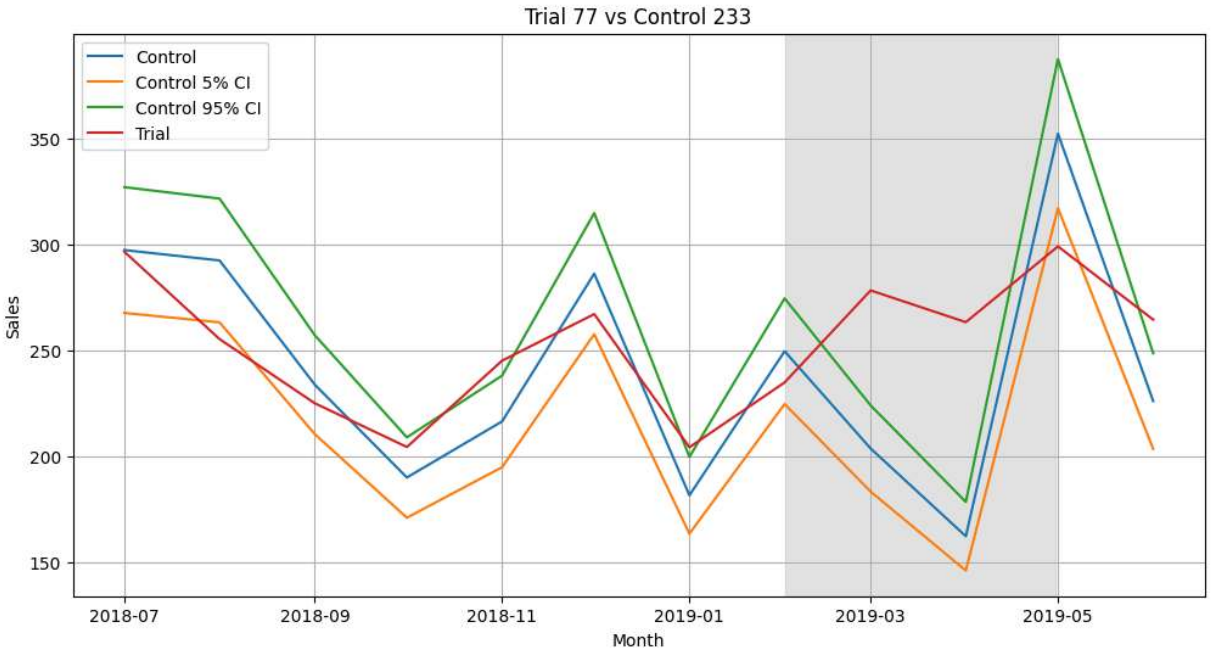
```
# Total sales pre-trial
trial_pre_sales = pre[pre["STORE_NBR"] == trial_store]["totSales"].sum()
control_pre_sales = pre[pre["STORE_NBR"] == control_store]["totSales"].sum()

scaling_factor = trial_pre_sales / control_pre_sales
print("\nScaling Factor:", scaling_factor)

df = measure_df.copy()

# Apply scaling
df["controlSales"] = np.where(
    df["STORE_NBR"] == control_store,
    df["totSales"] * scaling_factor,
    np.nan
)

# Compare trial vs scaled control
trial_df = df[df["STORE_NBR"] == trial_store][["YEARMONTH", "totSales"]]
control_df = df[df["STORE_NBR"] == control_store][["YEARMONTH", "controlSales"]

diff = trial_df.merge(control_df, on="YEARMONTH")
diff["perc_diff"] = abs(diff["totSales"] - diff["controlSales"]) / diff["contro

# Standard deviation pre-trial
std_dev = diff[diff["YEARMONTH"] < 201902]["perc_diff"].std()
print("Std Dev:", std_dev)

# Construct confidence intervals
control_series = control_df.rename(columns={"controlSales": "totSales"})
upper = control_series.copy()
lower = control_series.copy()

upper["totSales"] *= (1 + 2 * std_dev)
upper["Store_type"] = "Control 95% CI"

lower["totSales"] *= (1 - 2 * std_dev)
lower["Store_type"] = "Control 5% CI"

trial_plot = trial_df.rename(columns={"totSales": "totSales"})
trial_plot["Store_type"] = "Trial"

control_plot = control_series.copy()
control_plot["Store_type"] = "Control"

plot_df = pd.concat([trial_plot, control_plot, upper, lower])

plot_df["TransactionMonth"] = pd.to_datetime(
    plot_df["YEARMONTH"].astype(str) + "01", format="%Y%m%d"
)

# Plot
plt.figure(figsize=(12, 6))
for label, grp in plot_df.groupby("Store_type"):
    plt.plot(grp["TransactionMonth"], grp["totSales"], label=label)

# Shade trial period
```

```
    plt.axvspan(pd.to_datetime("20190201"), pd.to_datetime("20190430"),
                alpha=0.2, color="gray")

    plt.title(f"Trial {trial_store} vs Control {control_store}")
    plt.xlabel("Month")
    plt.ylabel("Sales")
    plt.grid(True)
    plt.legend()
    plt.show()

    return diff, std_dev
```

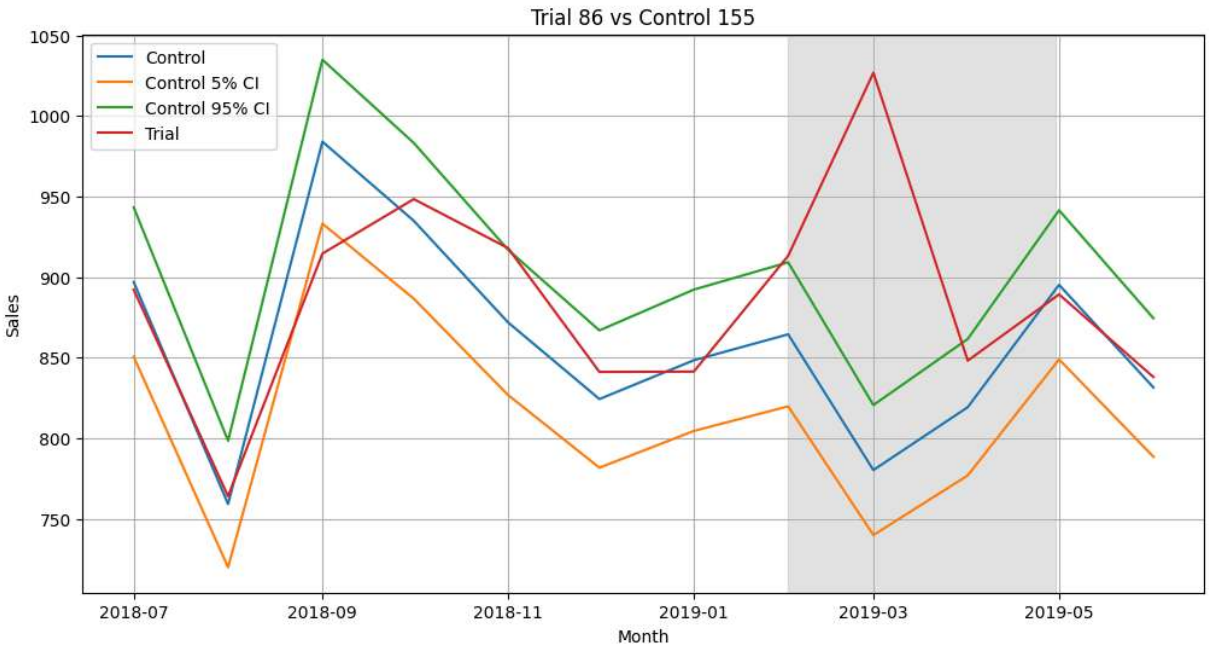
In [129…

```
#Apply Scaling
run_sales_uplift(measure_over_time, 77, 233)
run_sales_uplift(measure_over_time, 86, 155)
```

```
Scaling Factor: 1.023617303289553
Std Dev: 0.04994076264142537
```




```
Scaling Factor: 0.9700651481287743
Std Dev: 0.025833952854772368
```



Out[129…

	YEARMONTH	totSales	controlSales	perc_diff
0	201807	892.20	896.922236	0.005265
1	201808	764.05	759.269991	0.006296
2	201809	914.60	984.034086	0.070561
3	201810	948.40	934.948790	0.014387
4	201811	918.00	871.894555	0.052880
5	201812	841.20	824.361363	0.020426
6	201901	841.40	848.418979	0.008273
7	201902	913.20	864.522060	0.056306
8	201903	1026.80	780.320405	0.315870
9	201904	848.20	819.317024	0.035253
10	201905	889.30	895.224622	0.006618
11	201906	838.00	831.539845	0.007769,

np.float64(0.025833952854772368))

Interpretation:

Feb 2019 → +5.6% uplift (within 10% → not significant)

Mar 2019 → +31.6% uplift (significant)

Apr 2019 → +3.5% uplift (within 10% → not significant)

In [ ]:

In [ ]: