In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('QVI_data.csv')

data['DATE'] = pd.to_datetime(data['DATE'])

data.head()
```

Out[1]:

	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

In [2]:
```
# Creating a new column called YEARMONTH to turn a date like 2018-07-15 into 201807

#If I want to know if Store 77 is similar to Store 1, I can't just look at one Tuesday in July. Store 77 might have been busy that day because of a loca

data['YEARMONTH'] = data['DATE'].dt.year * 100 + data['DATE'].dt.month

data[['DATE', 'YEARMONTH']].head()
```

Out[2]:

	DATE	YEARMONTH
0	2018-10-17	201810
1	2018-09-16	201809
2	2019-03-07	201903
3	2019-03-08	201903
4	2018-11-02	201811

In [3]:
```
#  Grouping by Store and Month to get Sales and Customer counts
monthly_store_metrics = data.groupby(['STORE_NBR', 'YEARMONTH']).agg(
    totSales=('TOT_SALES', 'sum'),
    nCustomers=('LYLTY_CARD_NBR', 'nunique'),
    nTransactions=('TXN_ID', 'nunique')
).reset_index()

# Calculating Transactions per Customer (The "Loyalty" metric)
monthly_store_metrics['txnsPerCust'] = monthly_store_metrics['nTransactions'] / monthly_store_metrics['nCustomers']

monthly_store_metrics.head(10)
```

Out[3]:

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTransactions	txnsPerCust
0	1	201807	206.9	49	52	1.061224
1	1	201808	176.1	42	43	1.023810
2	1	201809	278.8	59	62	1.050847
3	1	201810	188.1	44	45	1.022727
4	1	201811	192.6	46	47	1.021739
5	1	201812	189.6	42	47	1.119048
6	1	201901	154.8	35	36	1.028571
7	1	201902	225.4	52	55	1.057692
8	1	201903	192.9	45	49	1.088889
9	1	201904	192.9	42	43	1.023810

In [4]:
```
# Counting how many months each store appears
store_counts = monthly_store_metrics.groupby('STORE_NBR').size()

# Looking for stores that appear 12 times
full_year_stores = store_counts[store_counts == 12].index
monthly_store_metrics = monthly_store_metrics[monthly_store_metrics['STORE_NBR'].isin(full_year_stores)]

print(f"We started with {len(store_counts)} stores, and now we have {len(full_year_stores)} reliable stores.")
```

```
We started with 272 stores, and now we have 260 reliable stores.
```

In [5]:
```
# Creating a table for only the months before the trial started
pre_trial_data = monthly_store_metrics[monthly_store_metrics['YEARMONTH'] < 201902]

# Making sure 201902 is not in there
pre_trial_data['YEARMONTH'].unique()
```

Out[5]:
```
array([201807, 201808, 201809, 201810, 201811, 201812, 201901],
      dtype=int32)
```

In [6]:
```
#  data for Store 77
trial_store_77 = pre_trial_data[pre_trial_data['STORE_NBR'] == 77]

# Store 77's sales month-to-month
trial_store_77[['YEARMONTH', 'totSales', 'nCustomers']]
```

Out[6]:

	YEARMONTH	totSales	nCustomers
880	201807	296.8	51
881	201808	255.5	47
882	201809	225.2	42
883	201810	204.5	37
884	201811	245.3	41
885	201812	267.3	46
886	201901	204.4	35

In [7]:
```
def find_store_score(trial_store_nbr, metric_col):
    #Calculating how similar other stores are to the trial store
    trial_data = pre_trial_data[pre_trial_data['STORE_NBR'] == trial_store_nbr][metric_col].values
    scores = []

    # Checking every other store
    for store in full_year_stores:
        if store == trial_store_nbr: continue

        potential_control = pre_trial_data[pre_trial_data['STORE_NBR'] == store][metric_col].values

        #  Calculating correlation
        corr = pd.Series(trial_data).corr(pd.Series(potential_control))

        # 2. Calculating the size
        dist = abs(trial_data - potential_control).mean()
        max_dist = pre_trial_data[metric_col].max() # Used for scaling
        mag = 1 - (dist / max_dist)

        scores.append({'STORE_NBR': store, 'Correlation': corr, 'Magnitude': mag})

    return pd.DataFrame(scores)
```

In [8]:
```
# To get scores for Sales and Customers
sales_scores = find_store_score(77, 'totSales')
cust_scores = find_store_score(77, 'nCustomers')

# Combining them for a final score
final_scores = sales_scores.merge(cust_scores, on='STORE_NBR', suffixes=('_Sales', '_Cust'))
final_scores['Final_Score'] = (final_scores['Correlation_Sales'] + final_scores['Correlation_Cust'] +
```

```
                                final_scores['Magnitude_Sales'] + final_scores['Magnitude_Cust']) / 4

final_scores.sort_values(by='Final_Score', ascending=False).head(5)
```

Out[8]:

	STORE_NBR	Correlation_Sales	Magnitude_Sales	Correlation_Cust	Magnitude_Cust	Final_Score
220	233	0.903774	0.988655	0.990358	0.995238	0.969506
38	41	0.783232	0.972678	0.844219	0.983810	0.895985
15	17	0.842668	0.906234	0.747308	0.976190	0.868100
240	254	0.577108	0.939108	0.916208	0.959048	0.847868
78	84	0.684348	0.867481	0.858571	0.950476	0.840219

In [9]:

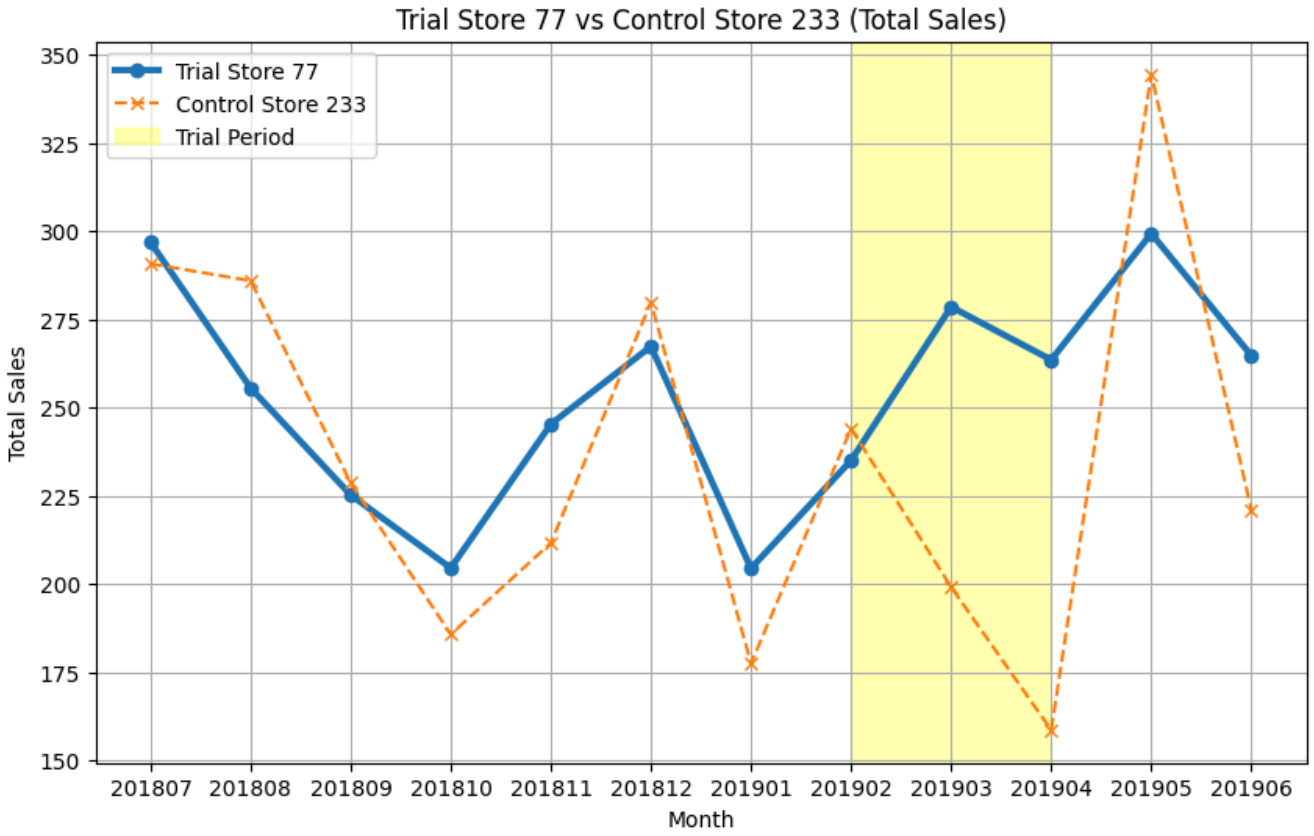
```
# A table with the trial and its twin
compare_sales = monthly_store_metrics[monthly_store_metrics['STORE_NBR'].isin([77, 233])]

# Pivoting the data so it's easy to plot
plot_data = compare_sales.pivot(index='YEARMONTH', columns='STORE_NBR', values='totSales')

#Plotting
plt.figure(figsize=(10, 6))
plt.plot(plot_data.index.astype(str), plot_data[77], label='Trial Store 77', marker='o', linewidth=3)
plt.plot(plot_data.index.astype(str), plot_data[233], label='Control Store 233', marker='x', linestyle='--')

# Highlighting the trial period
plt.axvspan('201902', '201904', color='yellow', alpha=0.3, label='Trial Period')

plt.title('Trial Store 77 vs Control Store 233 (Total Sales)')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.legend()
plt.grid(True)
plt.show()
```



In [10]:
```
# Store 86 score
sales_scores_86 = find_store_score(86, 'totSales')
cust_scores_86 = find_store_score(86, 'nCustomers')

#  Combining and finding the best match
final_scores_86 = sales_scores_86.merge(cust_scores_86, on='STORE_NBR', suffixes=('_Sales', '_Cust'))
final_scores_86['Final_Score'] = (final_scores_86['Correlation_Sales'] + final_scores_86['Correlation_Cust'] +
                                  final_scores_86['Magnitude_Sales'] + final_scores_86['Magnitude_Cust']) / 4

final_scores_86.sort_values(by='Final_Score', ascending=False).head(1)
```

Out[10]:

	STORE_NBR	Correlation_Sales	Magnitude_Sales	Correlation_Cust	Magnitude_Cust	Final_Score
146	155	0.877882	0.980438	0.942876	0.990476	0.947918

In [11]:
```
# Store 88 score
sales_scores_88 = find_store_score(88, 'totSales')
cust_scores_88 = find_store_score(88, 'nCustomers')
```

```
#  Combining and finding the best match
final_scores_88 = sales_scores_88.merge(cust_scores_88, on='STORE_NBR', suffixes=('_Sales', '_Cust'))
final_scores_88['Final_Score'] = (final_scores_88['Correlation_Sales'] + final_scores_88['Correlation_Cust'] +
                                  final_scores_88['Magnitude_Sales'] + final_scores_88['Magnitude_Cust']) / 4

final_scores_88.sort_values(by='Final_Score', ascending=False).head(1)
```

Out[11]:

	STORE_NBR	Correlation_Sales	Magnitude_Sales	Correlation_Cust	Magnitude_Cust	Final_Score
169	178	0.731857	0.755612	0.939466	0.852381	0.819829

In [12]:

```
# Creating a list of our pairs
pairs = [(77, 233), (86, 155), (88, 178)]
trial_months = [201902, 201903, 201904]

results = []

for trial, control in pairs:
    # Filtering data for trial months
    trial_sales = monthly_store_metrics[(monthly_store_metrics['STORE_NBR'] == trial) &
                                        (monthly_store_metrics['YEARMONTH'].isin(trial_months))]['totSales'].sum()

    control_sales = monthly_store_metrics[(monthly_store_metrics['STORE_NBR'] == control) &
                                          (monthly_store_metrics['YEARMONTH'].isin(trial_months))]['totSales'].sum()

    percentage_diff = ((trial_sales - control_sales) / control_sales) * 100
    results.append({'Trial_Store': trial, 'Control_Store': control, 'Sales_Lift_%': percentage_diff})

pd.DataFrame(results)
```

Out[12]:

	Trial_Store	Control_Store	Sales_Lift_%
0	77	233	29.134120
1	86	155	9.763011
2	88	178	36.201309

In [ ]:

In [ ]: