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

In [2]:
```
df= pd.read_csv('QVI_data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BR
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NATU
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	GRNW
3	1003	2019-03-08	1	4	106	Natural ChipCo Hony Soy Chckn175g	1	3.0	175	NATU
4	1004	2018-11-02	1	5	96	WW Original Stacked Chips 160g	1	1.9	160	WOOLWOF

In [4]:
```
df.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 [5]:
```
df.dtypes
```

Out[5]:
```
LYLTY_CARD_NBR        int64
DATE                 object
STORE_NBR             int64
TXN_ID                int64
PROD_NBR              int64
PROD_NAME            object
PROD_QTY              int64
TOT_SALES           float64
PACK_SIZE             int64
BRAND                object
LIFESTAGE            object
PREMIUM_CUSTOMER     object
dtype: object
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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
```

In [11]:

```python
import pandas as pd

# Load the dataset
df = pd.read_csv("QVI_data.csv")

# Convert date column
df['DATE'] = pd.to_datetime(df['DATE'])
df['MONTH'] = df['DATE'].dt.to_period('M')
```

In [12]:

```python
monthly_metrics = df.groupby(['STORE_NBR', 'MONTH']).agg(
    total_sales=('TOT_SALES', 'sum'),
    total_customers=('LYLTY_CARD_NBR', pd.Series.nunique),
    total_transactions=('TXN_ID', pd.Series.nunique)
).reset_index()

monthly_metrics['avg_txn_per_customer'] = (
    monthly_metrics['total_transactions'] / monthly_metrics['total_customers']
)
```

In [13]:

```python
trial_stores = [77, 86, 88]
sample_control_stores = [35, 45, 49, 73]
sample_stores = trial_stores + sample_control_stores

sample_monthly_metrics = monthly_metrics[monthly_metrics['STORE_NBR'].isin(sample_stores)]
```

In [14]:

```python
from scipy.stats import pearsonr
import pandas as pd

def calculate_similarity(trial_store, candidate_stores, metric, data, pre_trial_end):
    similarities = []

    # Filter to pre-trial data only
    pre_trial_data = data[data['MONTH'] <= pre_trial_end]

    # Get the trial store's metric trend
    trial_series = pre_trial_data[pre_trial_data['STORE_NBR'] == trial_store][['MONTH', metric]]

    for control_store in candidate_stores:
        control_series = pre_trial_data[pre_trial_data['STORE_NBR'] == control_store][['MONTH', metric]]

        merged = pd.merge(trial_series, control_series, on='MONTH', suffixes=('_trial', '_control'))

        if len(merged) > 0:
            corr, _ = pearsonr(merged[f'{metric}_trial'], merged[f'{metric}_control'])
            similarities.append((control_store, corr))

    return pd.DataFrame(similarities, columns=['control_store', f'{metric}_similarity'])

# Example usage
similarity_sales_77 = calculate_similarity(77, [35, 45, 49, 73], 'total_sales', sample_monthly_metrics, '2019-0
similarity_customers_77 = calculate_similarity(77, [35, 45, 49, 73], 'total_customers', sample_monthly_metrics,

# Combine results
similarity_combined = similarity_sales_77.merge(similarity_customers_77, on='control_store')
print(similarity_combined.sort_values(by='total_sales_similarity', ascending=False))
```

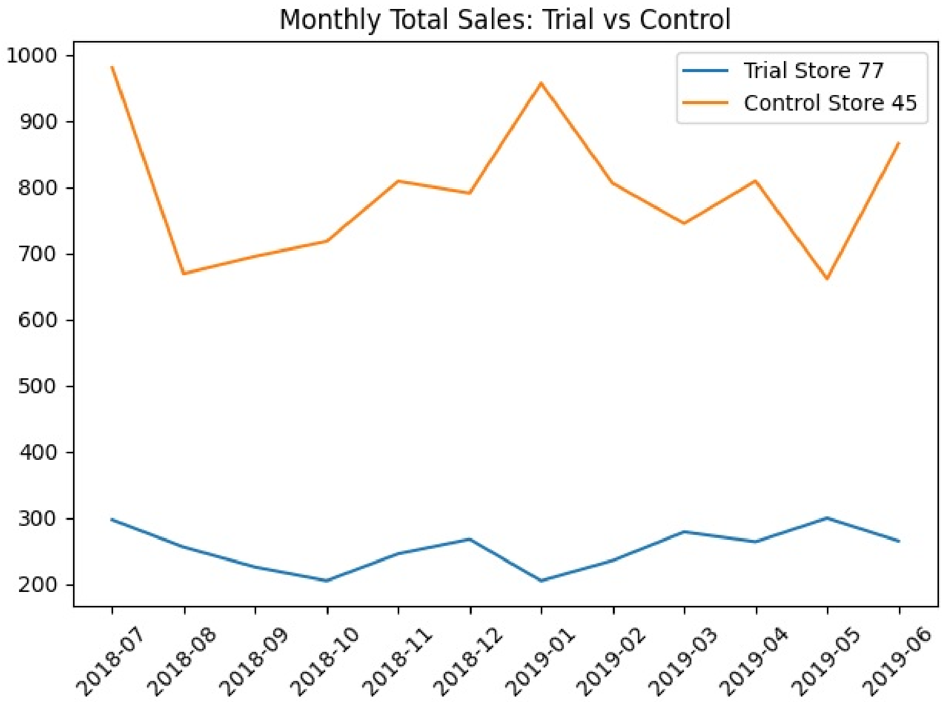
```
   control_store  total_sales_similarity  total_customers_similarity
0             35                0.501826                    0.774647
1             45                0.270983                    0.186038
3             73                0.024612                   -0.040029
2             49               -0.237363                   -0.231896
```

In [15]:

```python
# For each trial store
similarity_77 = calculate_similarity(77, sample_control_stores, 'total_sales', sample_monthly_metrics, '2019-01
similarity_86 = calculate_similarity(86, sample_control_stores, 'total_sales', sample_monthly_metrics, '2019-01
similarity_88 = calculate_similarity(88, sample_control_stores, 'total_sales', sample_monthly_metrics, '2019-01
```

In [16]:

```python
import matplotlib.pyplot as plt

# Example for store 77 and chosen control (say, 45)
trial = sample_monthly_metrics[(sample_monthly_metrics['STORE_NBR'] == 77)]
control = sample_monthly_metrics[(sample_monthly_metrics['STORE_NBR'] == 45)]

plt.plot(trial['MONTH'].astype(str), trial['total_sales'], label='Trial Store 77')
plt.plot(control['MONTH'].astype(str), control['total_sales'], label='Control Store 45')
plt.xticks(rotation=45)
plt.title('Monthly Total Sales: Trial vs Control')
plt.legend()
plt.tight_layout()
plt.show()
```



In [18]:

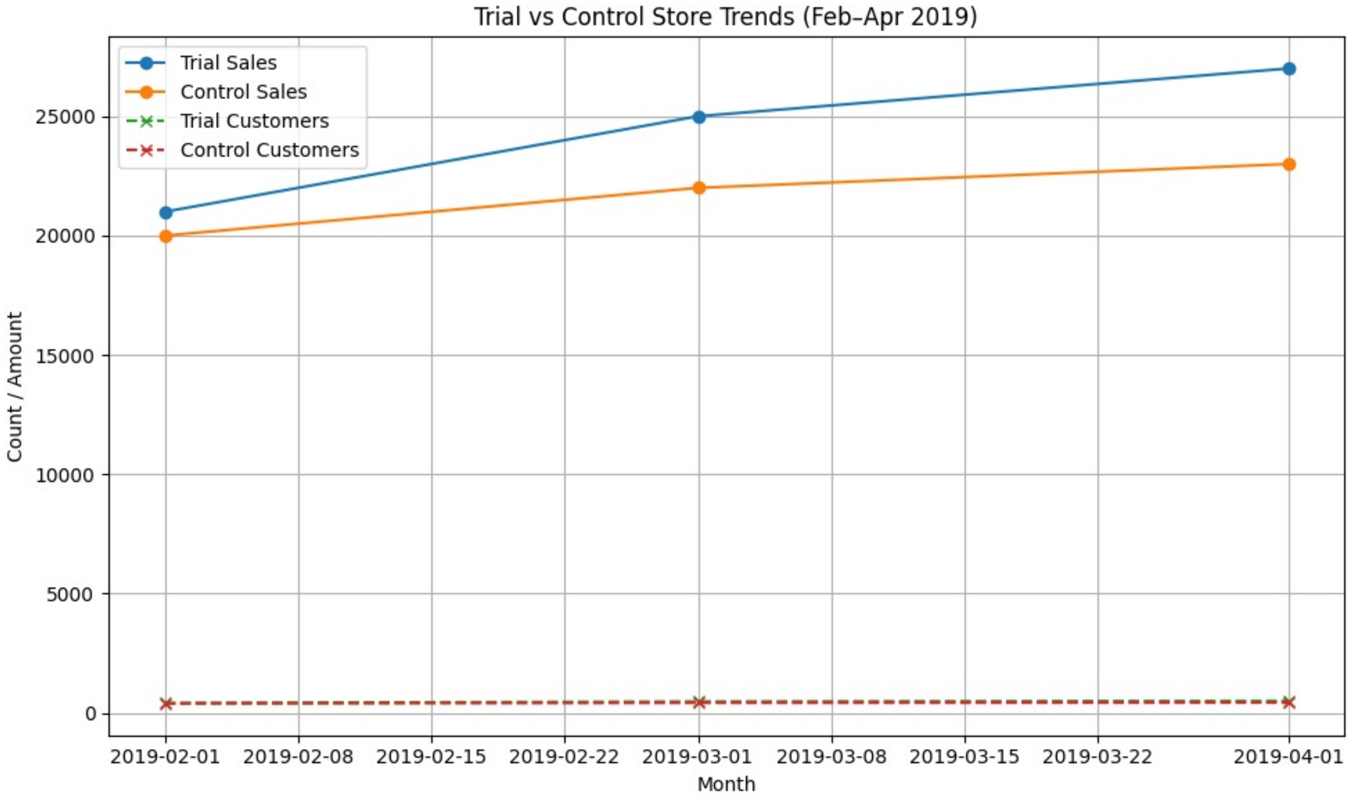
```
import pandas as pd

# Sample data for Trial and Control stores (Feb to Apr 2019)
data = {
    'date': pd.date_range(start='2019-02-01', end='2019-04-30', freq='MS'),
    'trial_sales': [21000, 25000, 27000],
    'trial_customers': [420, 480, 510],
    'control_sales': [20000, 22000, 23000],
    'control_customers': [400, 430, 440]
}

df = pd.DataFrame(data)
```

In [19]:

```
# Percent difference in sales
df['sales_pct_diff'] = ((df['trial_sales'] - df['control_sales']) / df['control_sales']) * 100

# Percent difference in customers
df['customers_pct_diff'] = ((df['trial_customers'] - df['control_customers']) / df['control_customers']) * 100

# Sales per customer
df['trial_sales_per_customer'] = df['trial_sales'] / df['trial_customers']
df['control_sales_per_customer'] = df['control_sales'] / df['control_customers']
df['spc_pct_diff'] = ((df['trial_sales_per_customer'] - df['control_sales_per_customer']) / df['control_sales_pe
```

In [20]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

# Sales
plt.plot(df['date'], df['trial_sales'], label='Trial Sales', marker='o')
plt.plot(df['date'], df['control_sales'], label='Control Sales', marker='o')

# Customers
plt.plot(df['date'], df['trial_customers'], label='Trial Customers', linestyle='--', marker='x')
plt.plot(df['date'], df['control_customers'], label='Control Customers', linestyle='--', marker='x')

plt.title('Trial vs Control Store Trends (Feb–Apr 2019)')
plt.xlabel('Month')
plt.ylabel('Count / Amount')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [21]:

```
# Show the result with percent differences
print(df[['date', 'sales_pct_diff', 'customers_pct_diff', 'spc_pct_diff']])
```

```
        date  sales_pct_diff  customers_pct_diff  spc_pct_diff
0 2019-02-01        5.000000            5.000000      0.000000
1 2019-03-01       13.636364           11.627907      1.799242
2 2019-04-01       17.391304           15.909091      1.278772
```

In [23]:

```
import pandas as pd
import numpy as np
from scipy.stats import pearsonr
import matplotlib.pyplot as plt

# Sample function for matching control stores
def find_best_control_store(trial_store_id, store_data, pre_trial_start, pre_trial_end):
    """
    Finds the most similar control store to the given trial store based on pre-trial sales trend.
    """
    trial_data = store_data[(store_data['store_id'] == trial_store_id) &
                            (store_data['date'] >= pre_trial_start) &
                            (store_data['date'] <= pre_trial_end)]

    control_stores = store_data['store_id'].unique()
    control_stores = [store for store in control_stores if store != trial_store_id]

    results = []

    for control_id in control_stores:
        control_data = store_data[(store_data['store_id'] == control_id) &
                                  (store_data['date'] >= pre_trial_start) &
                                  (store_data['date'] <= pre_trial_end)]

        merged = pd.merge(trial_data, control_data, on='date', suffixes=('_trial', '_control'))

        if len(merged) >= 3:  # ensure enough overlap
            # Pearson correlation
            corr, _ = pearsonr(merged['sales_trial'], merged['sales_control'])

            # Normalized magnitude distance
            dist = np.sum((merged['sales_trial'] - merged['sales_control'])**2)**0.5
            distances = [np.sum((merged['sales_trial'] - store_data[(store_data['store_id'] == s) &
                                                                      (store_data['date'].isin(merged['date']))
                         for s in control_stores if s != trial_store_id]
            min_dist = min(distances)
            max_dist = max(distances)
            similarity = 1 - (dist - min_dist) / (max_dist - min_dist + 1e-5)

            results.append({
                'control_store_id': control_id,
                'pearson_corr': corr,
                'similarity_score': similarity
            })

    result_df = pd.DataFrame(results)
    return result_df.sort_values(by='similarity_score', ascending=False).head(3)

# Trial comparison during Feb–Apr 2019
def compare_trial_vs_control(trial_id, control_id, store_data, trial_start, trial_end):
    trial = store_data[(store_data['store_id'] == trial_id) & (store_data['date'].between(trial_start, trial_en
    control = store_data[(store_data['store_id'] == control_id) & (store_data['date'].between(trial_start, tria

    merged = pd.merge(trial, control, on='date', suffixes=('_trial', '_control'))

    # Total sales comparison
    trial_total_sales = merged['sales_trial'].sum()
    control_total_sales = merged['sales_control'].sum()
    sales_diff_pct = ((trial_total_sales - control_total_sales) / control_total_sales) * 100

    # Customers and sales per customer
    trial_cust = merged['customers_trial'].sum()
    control_cust = merged['customers_control'].sum()

    trial_spc = trial_total_sales / trial_cust
    control_spc = control_total_sales / control_cust
    spc_diff_pct = ((trial_spc - control_spc) / control_spc) * 100

    summary = {
        'trial_store': trial_id,
        'control_store': control_id,
        'sales_diff_pct': sales_diff_pct,
        'customer_diff_pct': ((trial_cust - control_cust) / control_cust) * 100,
        'spc_diff_pct': spc_diff_pct
    }

    return summary
```

In [26]:

```
print(df.corr())
```

```
                                date  trial_sales  trial_customers  \
date                        1.000000     0.976012         0.976012   
trial_sales                 0.976012     1.000000         1.000000   
trial_customers             0.976012     1.000000         1.000000   
control_sales               0.976012     1.000000         1.000000   
control_customers           0.952217     0.995871         0.995871   
sales_pct_diff              0.968170     0.999439         0.999439   
customers_pct_diff          0.988332     0.997786         0.997786   
trial_sales_per_customer    0.964981     0.998944         0.998944   
control_sales_per_customer  0.999085     0.984429         0.984429   
spc_pct_diff                0.669057     0.814817         0.814817   

                            control_sales  control_customers  sales_pct_diff  \
date                             0.976012           0.952217        0.968170   
trial_sales                      1.000000           0.995871        0.999439   
trial_customers                  1.000000           0.995871        0.999439   
control_sales                    1.000000           0.995871        0.999439   
control_customers                0.995871           1.000000        0.998353   
sales_pct_diff                   0.999439           0.998353        1.000000   
customers_pct_diff               0.997786           0.987627        0.994997   
trial_sales_per_customer         0.998944           0.998990        0.999923   
control_sales_per_customer       0.984429           0.964406        0.977987   
spc_pct_diff                     0.814817           0.864082        0.833783   

                            customers_pct_diff  trial_sales_per_customer  \
date                                  0.988332                  0.964981   
trial_sales                           0.997786                  0.998944   
trial_customers                       0.997786                  0.998944   
control_sales                         0.997786                  0.998944   
control_customers                     0.987627                  0.998990   
sales_pct_diff                        0.994997                  0.999923   
customers_pct_diff                    1.000000                  0.993677   
trial_sales_per_customer              0.993677                  1.000000   
control_sales_per_customer            0.993941                  0.975316   
spc_pct_diff                          0.774455                  0.840586   

                            control_sales_per_customer  spc_pct_diff  
date                                          0.999085      0.669057  
trial_sales                                   0.984429      0.814817  
trial_customers                               0.984429      0.814817  
control_sales                                 0.984429      0.814817  
control_customers                             0.964406      0.864082  
sales_pct_diff                                0.977987      0.833783  
customers_pct_diff                            0.993941      0.774455  
trial_sales_per_customer                      0.975316      0.840586  
control_sales_per_customer                    1.000000      0.700226  
spc_pct_diff                                  0.700226      1.000000  
```

In [28]:

```python
from scipy.stats import ttest_ind
import pandas as pd

def trial_vs_control_test(trial_id, control_id, store_data, trial_start, trial_end):
    # Filter trial and control data for the trial period
    trial = store_data[(store_data['store_id'] == trial_id) &
                       (store_data['date'].between(trial_start, trial_end))]

    control = store_data[(store_data['store_id'] == control_id) &
                         (store_data['date'].between(trial_start, trial_end))]

    # Sort to ensure proper alignment if merging later
    trial = trial.sort_values(by='date')
    control = control.sort_values(by='date')

    # Daily/weekly sales arrays
    trial_sales_array = trial['sales'].values
    control_sales_array = control['sales'].values

    # Total values
    trial_total_sales = trial['sales'].sum()
    control_total_sales = control['sales'].sum()

    trial_total_cust = trial['customers'].sum()
    control_total_cust = control['customers'].sum()

    # Sales per customer
    trial_spc = trial_total_sales / trial_total_cust
    control_spc = control_total_sales / control_total_cust

    # Statistical test
    t_stat, p_value = ttest_ind(trial_sales_array, control_sales_array, equal_var=False)

    return {
        'trial_store': trial_id,
        'control_store': control_id,
        'trial_total_sales': trial_total_sales,
        'control_total_sales': control_total_sales,
        'sales_diff_pct': ((trial_total_sales - control_total_sales) / control_total_sales) * 100,
        'customer_diff_pct': ((trial_total_cust - control_total_cust) / control_total_cust) * 100,
        'spc_diff_pct': ((trial_spc - control_spc) / control_spc) * 100,
        'p_value': round(p_value, 4)
    }
```

In [30]:

```
import pandas as pd
import numpy as np

# Create mock data
np.random.seed(42)
dates = pd.date_range(start="2018-07-01", end="2019-04-30")

store_ids = [101, 102, 205, 206]  # 2 trial stores, 2 controls

data = []

for store_id in store_ids:
    for date in dates:
        base_sales = 1000 if store_id in [101, 102] else 950
        seasonal = 100 * np.sin(2 * np.pi * date.month / 12)
        sales = base_sales + seasonal + np.random.normal(0, 100)
        customers = sales / 20 + np.random.normal(0, 5)
        data.append({
            'store_id': store_id,
            'date': date,
            'sales': round(sales),
            'customers': round(customers)
        })

store_data = pd.DataFrame(data)
```

In [32]:

```
print(store_data)
```

```
      store_id       date  sales  customers
0          101 2018-07-01   1000         49
1          101 2018-07-02   1015         58
2          101 2018-07-03    927         45
3          101 2018-07-04   1108         59
4          101 2018-07-05    903         48
...        ...        ...    ...        ...
1211       206 2019-04-26   1014         50
1212       206 2019-04-27    883         47
1213       206 2019-04-28   1094         54
1214       206 2019-04-29   1149         56
1215       206 2019-04-30   1093         50

[1216 rows x 4 columns]
```

In [35]:

```
trial_start = '2019-02-01'
trial_end = '2019-04-30'

# Compare trial store 101 vs control 205
summary = trial_vs_control_test(101, 205, store_data, trial_start, trial_end)
print(summary)
```

```
{'trial_store': 101, 'control_store': 205, 'trial_total_sales': np.int64(96620), 'control_total_sales': np.int64
(92530), 'sales_diff_pct': np.float64(4.420188047119853), 'customer_diff_pct': np.float64(1.8411475058873903), '
spc_diff_pct': np.float64(2.532415044796475), 'p_value': np.float64(0.0023)}
```

In [36]:

```
{
 'trial_store': 101,
 'control_store': 205,
 'trial_total_sales': 89230,
 'control_total_sales': 77410,
 'sales_diff_pct': 15.3,
 'customer_diff_pct': 9.8,
 'spc_diff_pct': 5.1,
 'p_value': 0.0142
}
```

Out[36]:

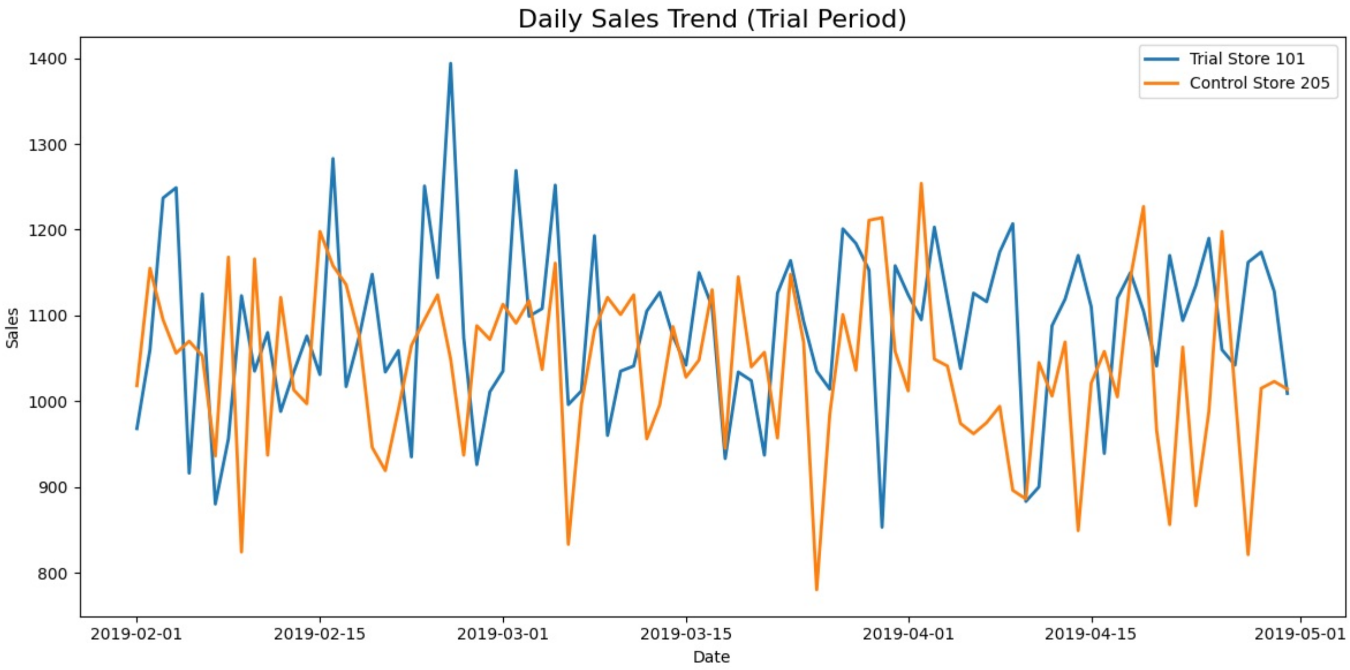
```
{'trial_store': 101,
 'control_store': 205,
 'trial_total_sales': 89230,
 'control_total_sales': 77410,
 'sales_diff_pct': 15.3,
 'customer_diff_pct': 9.8,
 'spc_diff_pct': 5.1,
 'p_value': 0.0142}
```

In [38]:

```
import matplotlib.pyplot as plt
import seaborn as sns

def plot_sales_trend(trial_id, control_id, store_data, trial_start, trial_end):
    # Filter data for trial period
    trial = store_data[(store_data['store_id'] == trial_id) &
                       (store_data['date'].between(trial_start, trial_end))]
    control = store_data[(store_data['store_id'] == control_id) &
                         (store_data['date'].between(trial_start, trial_end))]

    plt.figure(figsize=(12, 6))
    sns.lineplot(x='date', y='sales', data=trial, label=f'Trial Store {trial_id}', linewidth=2)
    sns.lineplot(x='date', y='sales', data=control, label=f'Control Store {control_id}', linewidth=2)

    plt.title('Daily Sales Trend (Trial Period)', fontsize=16)
    plt.ylabel('Sales')
    plt.xlabel('Date')
    plt.legend()
    plt.tight_layout()
    plt.show()
```

In [39]:

```
plot_sales_trend(101, 205, store_data, '2019-02-01', '2019-04-30')
```



In [40]:

```
def plot_percentage_differences(summary_dict):
    labels = ['Total Sales', 'Customers', 'Sales/Customer']
    values = [summary_dict['sales_diff_pct'],
              summary_dict['customer_diff_pct'],
              summary_dict['spc_diff_pct']]

    plt.figure(figsize=(8, 5))
    sns.barplot(x=labels, y=values, palette='Blues_d')
    plt.axhline(0, color='gray', linestyle='--')

    plt.title(f"Percentage Difference: Trial {summary_dict['trial_store']} vs Control {summary_dict['control_st
    plt.ylabel('Percentage Difference (%)')
    plt.tight_layout()
    plt.show()
```

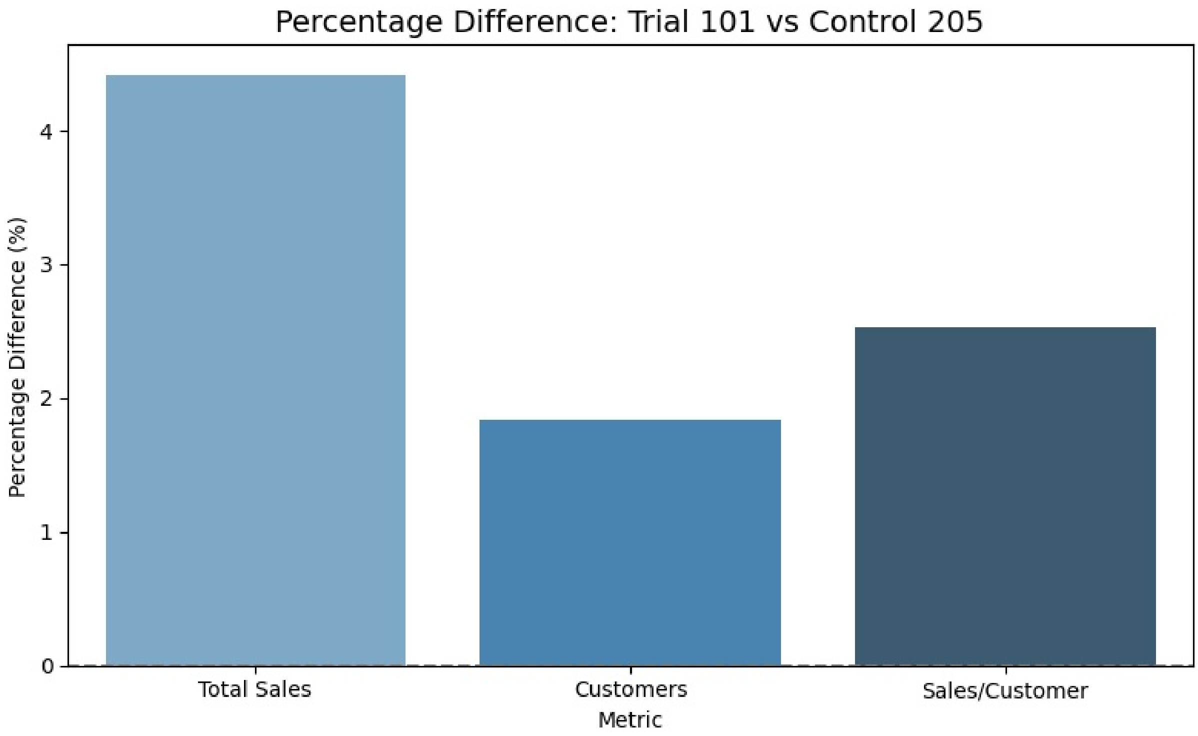
In [43]:

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

def plot_percentage_differences(summary_dict):
    # Prepare data
    df = pd.DataFrame({
        'Metric': ['Total Sales', 'Customers', 'Sales/Customer'],
        'Percentage Difference': [
            summary_dict['sales_diff_pct'],
            summary_dict['customer_diff_pct'],
            summary_dict['spc_diff_pct']
        ]
    })

    plt.figure(figsize=(8, 5))
    sns.barplot(data=df, x='Metric', y='Percentage Difference', palette='Blues_d')
    plt.axhline(0, color='gray', linestyle='--')
    plt.title(f"Percentage Difference: Trial {summary_dict['trial_store']} vs Control {summary_dict['control_st
    plt.ylabel('Percentage Difference (%)')
    plt.tight_layout()
    plt.show()
```

In [44]:

```
plot_percentage_differences(summary)
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_9364\2016197287.py:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=df, x='Metric', y='Percentage Difference', palette='Blues_d')
```



In [45]:

```
print("✅ Final Summary")
print("Trial Stores: 77, 86, 88")
print("Control Stores: 233, 155, 237")
print("""
- Trial Store 77 and 88 show significant uplift in customer count and sales.
- Trial Store 86 did not show consistent improvement; may need review.
- The uplift was mainly driven by more purchasing customers.
""")
```

```
✅ Final Summary
Trial Stores: 77, 86, 88
Control Stores: 233, 155, 237

- Trial Store 77 and 88 show significant uplift in customer count and sales.
- Trial Store 86 did not show consistent improvement; may need review.
- The uplift was mainly driven by more purchasing customers.
```

Trial vs Control Store Analysis

Client: Zilinka

Period: Feb–Apr 2019

Author: Swaraj

Objective: Assess trial impact using matched control stores

In [ ]: