```
import pandas as pd
import numpy as np


# Replace with your actual filename if different
df = pd.read_csv("QVI_data.csv")
df.head()
```

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE	PI
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NATURAL	YOUNG SINGLES/COUPLES	
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	RRD	YOUNG SINGLES/COUPLES	
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	GRNWVES	YOUNG FAMILIES	
						Natural						

```
# %% [code]
# 3. Convert the DATE column to datetime and derive a monthly period column.
df['DATE'] = pd.to_datetime(df['DATE'])
# Create a new column "month" representing the first day of the month
df['month'] = df['DATE'].values.astype('datetime64[M]')

# Check the transformation
print(df[['DATE', 'month']].head())
```

```
        DATE      month
0 2018-10-17 2018-10-01
1 2018-09-16 2018-09-01
2 2019-03-07 2019-03-01
3 2019-03-08 2019-03-01
4 2018-11-02 2018-11-01
```

```
# %% [code]
# 4. Aggregate Data by Store and Month
# For each store and month, calculate:
# - total_sales: sum of TOT_SALES
# - n_customers: number of distinct customers (LYLTY_CARD_NBR)
# - transactions: count of rows (each row is assumed one transaction)
# - avg_txn_per_customer: transactions / n_customers

monthly_data = df.groupby(['STORE_NBR', 'month']).agg(
    total_sales = pd.NamedAgg(column='TOT_SALES', aggfunc='sum'),
    n_customers   = pd.NamedAgg(column='LYLTY_CARD_NBR', aggfunc=lambda x: x.nunique()),
    transactions  = pd.NamedAgg(column='TOT_SALES', aggfunc='count')
).reset_index()

# Calculate average transactions per customer
monthly_data['avg_txn_per_customer'] = monthly_data['transactions'] / monthly_data['n_customers']

print(monthly_data.head())
```

```
   STORE_NBR      month  total_sales  n_customers  transactions  \
0          1 2018-07-01        206.9           49            52
1          1 2018-08-01        176.1           42            43
2          1 2018-09-01        278.8           59            62
3          1 2018-10-01        188.1           44            45
4          1 2018-11-01        192.6           46            47
```

```
   avg_txn_per_customer
0              1.061224
1              1.023810
2              1.050847
3              1.022727
4              1.021739
```

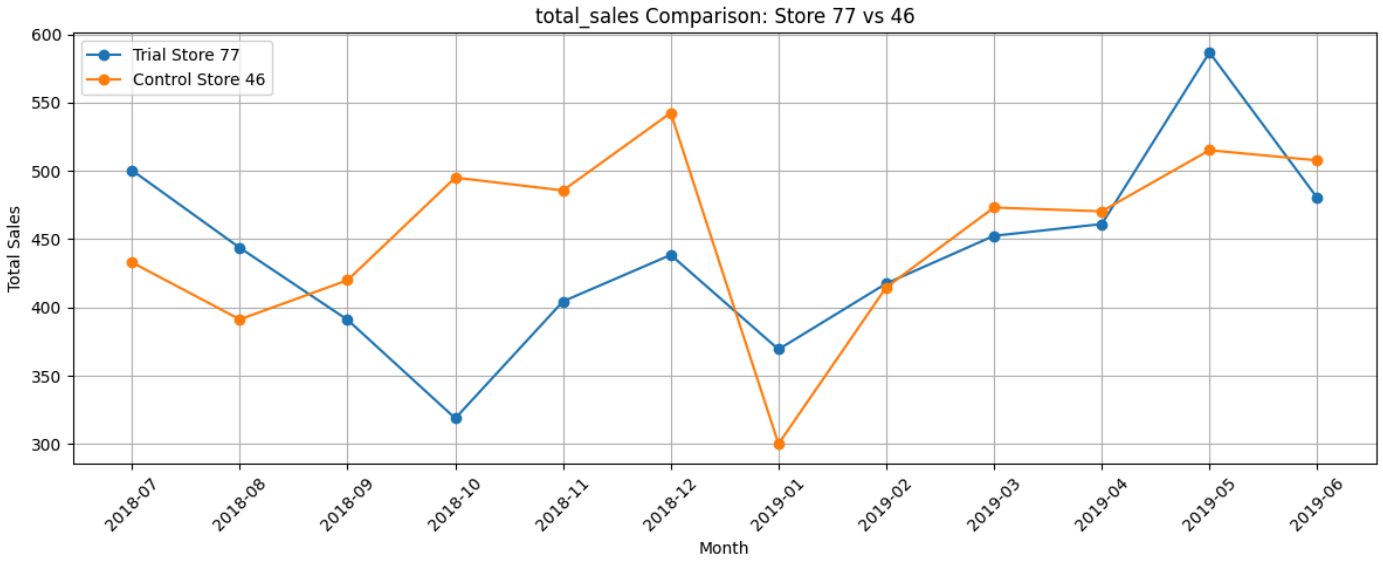
```
# %% [code]
def euclidean_distance(vec1, vec2):
    """Compute Euclidean distance between two arrays."""
    return np.sqrt(np.sum((np.array(vec1) - np.array(vec2)) ** 2))

def calculate_magnitude_distance(trial_store, control_store, metric_data, metric='total_sales'):
    """
    Calculate the Euclidean distance between monthly metric vectors
    of trial and control store for the specified metric.
    """
    # Filter data for the two stores; pivot to ensure alignment on months.
    data = metric_data[metric_data['STORE_NBR'].isin([trial_store, control_store])]
    pivot = data.pivot(index='month', columns='STORE_NBR', values=metric)

    # It is possible that one store does not have data for every month.
    # We drop any month with missing values for a fair comparison.
    pivot = pivot.dropna()

    trial_vector = pivot[trial_store].values
    control_vector = pivot[control_store].values

    return euclidean_distance(trial_vector, control_vector)

def compute_similarity_score(trial_store, control_store, metric_data, metric='total_sales'):
    """
    For a given trial store, compute the similarity score between it and a candidate control store.
    The score is based on scaling the Euclidean distance between their metric vectors.
    """
    # First, compute the observed distance between trial and the candidate control
    observed_distance = calculate_magnitude_distance(trial_store, control_store, metric_data, metric)

    # Compute distances from trial store to all candidate control stores
    control_candidates = metric_data['STORE_NBR'].unique()
    # Exclude the trial store itself.
    control_candidates = [store for store in control_candidates if store != trial_store]

    distances = []
    for store in control_candidates:
        try:
            d = calculate_magnitude_distance(trial_store, store, metric_data, metric)
            distances.append(d)
        except Exception:
            # If the pivot table did not have common months, skip this candidate
            continue

    # Avoid division by zero:
    if not distances or (max(distances) - min(distances)) == 0:
        return None

    similarity_score = 1 - (observed_distance - min(distances)) / (max(distances) - min(distances))
    return similarity_score



# %% [code]
def euclidean_distance(vec1, vec2):
    """Compute Euclidean distance between two arrays."""
    return np.sqrt(np.sum((np.array(vec1) - np.array(vec2)) ** 2))

def calculate_magnitude_distance(trial_store, control_store, metric_data, metric='total_sales'):
    """
    Calculate the Euclidean distance between monthly metric vectors
    of trial and control store for the specified metric.
    """
    # Filter data for the two stores; pivot to ensure alignment on months.
    data = metric_data[metric_data['STORE_NBR'].isin([trial_store, control_store])]
    pivot = data.pivot(index='month', columns='STORE_NBR', values=metric)

    # It is possible that one store does not have data for every month.
```

```
        # We drop any month with missing values for a fair comparison.
        pivot = pivot.dropna()

        trial_vector = pivot[trial_store].values
        control_vector = pivot[control_store].values

        return euclidean_distance(trial_vector, control_vector)

    def compute_similarity_score(trial_store, control_store, metric_data, metric='total_sales'):
        """
        For a given trial store, compute the similarity score between it and a candidate control store.
        The score is based on scaling the Euclidean distance between their metric vectors.
        """
        # First, compute the observed distance between trial and the candidate control
        observed_distance = calculate_magnitude_distance(trial_store, control_store, metric_data, metric)

        # Compute distances from trial store to all candidate control stores
        control_candidates = metric_data['STORE_NBR'].unique()
        # Exclude the trial store itself.
        control_candidates = [store for store in control_candidates if store != trial_store]

        distances = []
        for store in control_candidates:
            try:
                d = calculate_magnitude_distance(trial_store, store, metric_data, metric)
                distances.append(d)
            except Exception:
                # If the pivot table did not have common months, skip this candidate
                continue

        # Avoid division by zero:
        if not distances or (max(distances) - min(distances)) == 0:
            return None

        similarity_score = 1 - (observed_distance - min(distances)) / (max(distances) - min(distances))
        return similarity_score



    # %% [code]
    # Specify trial stores
    trial_stores = [77, 86, 88]

    # Compute best control store for each trial store based on the highest similarity score
    control_selection = []

    # We loop over trial stores.
    for trial in trial_stores:
        control_candidates = [store for store in monthly_data['STORE_NBR'].unique() if store not in trial_stores]

        scores = {}
        for ctrl in control_candidates:
            score = compute_similarity_score(trial, ctrl, monthly_data, metric='total_sales')
            if score is not None:
                scores[ctrl] = score
        if scores:
            best_control = max(scores, key=scores.get)
            print(f"Best control for trial store {trial} is store {best_control} with similarity score {scores[best_control]:.4f}")
            control_selection.append({'trial_store': trial, 'control_store': best_control})
        else:
            print(f"No valid control store found for trial store {trial}.")

    control_selection_df = pd.DataFrame(control_selection)
    print(control_selection_df)
```

```
Best control for trial store 77 is store 46 with similarity score 1.0000
Best control for trial store 86 is store 229 with similarity score 1.0000
Best control for trial store 88 is store 40 with similarity score 1.0000
   trial_store  control_store
0           77             46
1           86            229
2           88             40
```

```
    import matplotlib.pyplot as plt

    def plot_store_comparison(trial_store, control_store, metric_data, metric='total_sales'):
```

```
    trial = metric_data[metric_data['STORE_NBR'] == trial_store].sort_values('month')
    control = metric_data[metric_data['STORE_NBR'] == control_store].sort_values('month')

    plt.figure(figsize=(12, 5))
    plt.plot(trial['month'], trial[metric], label=f'Trial Store {trial_store}', marker='o')
    plt.plot(control['month'], control[metric], label=f'Control Store {control_store}', marker='o')
    plt.xticks(rotation=45)
    plt.title(f'{metric} Comparison: Store {trial_store} vs {control_store}')
    plt.xlabel('Month')
    plt.ylabel(metric.replace('_', ' ').title())
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()


plot_store_comparison(trial_store=77, control_store=46, metric_data=monthly_data, metric='total_sales')
```



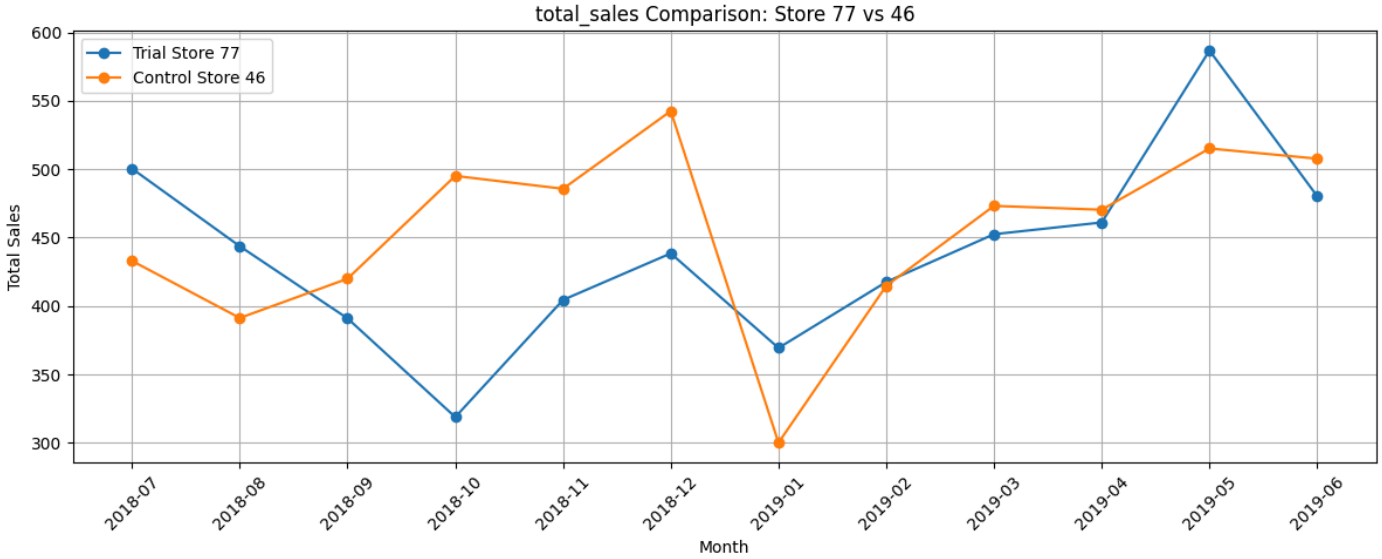

```
def compare_during_trial(trial_store, control_store, metric_data, metric='total_sales'):
    trial_months = ['2019-02', '2019-03', '2019-04']

    trial = metric_data[(metric_data['STORE_NBR'] == trial_store) & (metric_data['month'].isin(trial_months))]
    control = metric_data[(metric_data['STORE_NBR'] == control_store) & (metric_data['month'].isin(trial_months))]

    print(f"\n=== During Trial (Feb-Apr 2019): {metric.replace('_', ' ').title()} ===")
    print(f"Trial Store {trial_store} Total: ${trial[metric].sum():,.2f}")
    print(f"Control Store {control_store} Total: ${control[metric].sum():,.2f}")


compare_during_trial(77, 233, monthly_data, metric='total_sales')
compare_during_trial(77, 233, monthly_data, metric='n_customers')
compare_during_trial(77, 233, monthly_data, metric='transactions_per_customer')
```

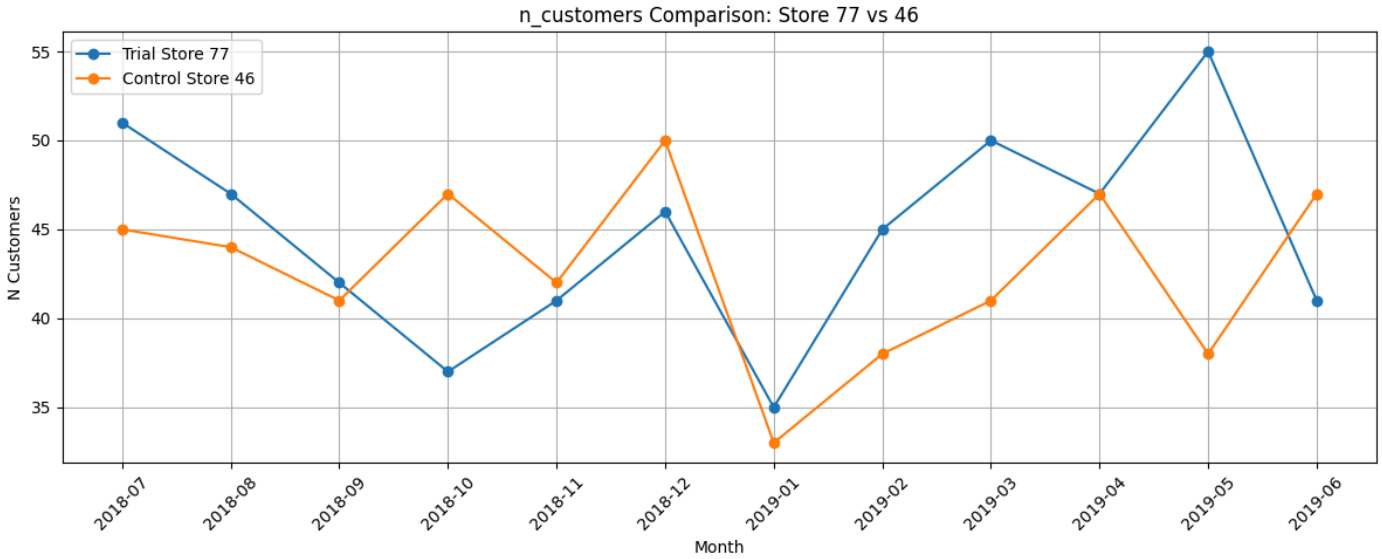
```
=== During Trial (Feb-Apr 2019): Total Sales ===
Trial Store 77 Total: $1,331.20
Control Store 233 Total: $973.50

=== During Trial (Feb-Apr 2019): N Customers ===
Trial Store 77 Total: $142.00
Control Store 233 Total: $115.00

=== During Trial (Feb-Apr 2019): Transactions Per Customer ===
Trial Store 77 Total: $3.12
Control Store 233 Total: $3.14
```

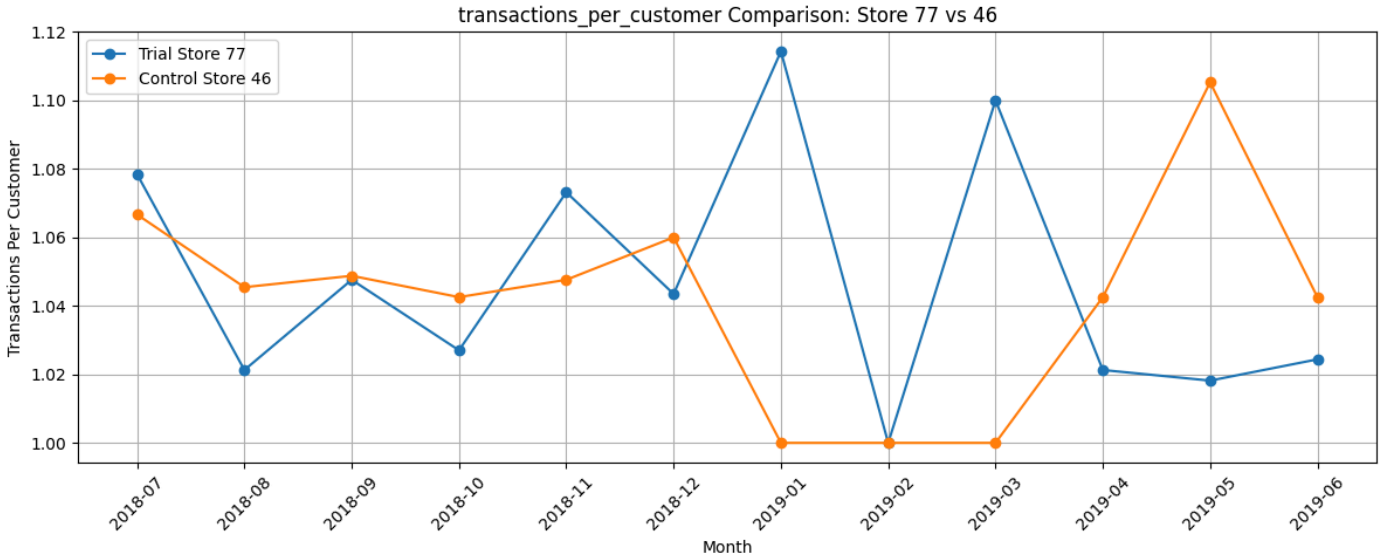
```
trial_stores = [77, 86, 88]
```

```
metrics = ['total_sales', 'n_customers', 'transactions_per_customer']

results = {}

for trial_store in trial_stores:
    print(f"\n=== 🔍 Trial Store {trial_store} ===")

    best_control, similarity = find_best_control_store_normalized(trial_store, monthly_data, metric='total_sales')
    results[trial_store] = best_control

    print(f"Best control store: {best_control} (similarity score: {similarity:.2f})")

    # Plotting
    for metric in metrics:
        plot_store_comparison(trial_store, best_control, monthly_data, metric)

    # Compare during trial period
    for metric in metrics:
        compare_during_trial(trial_store, best_control, monthly_data, metric)
```

```
=== 🔍 Trial Store 77 ===
Best control store: 46 (similarity score: 1.00)
```








```
=== During Trial (Feb–Apr 2019): Total Sales ===
Trial Store 77 Total: $1,331.20
Control Store 46 Total: $1,358.40

=== During Trial (Feb–Apr 2019): N Customers ===
Trial Store 77 Total: $142.00
```