```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr
from sklearn.metrics.pairwise import euclidean_distances
```

```
# Load the dataset (make sure QVI_data.csv is in your working directory)
df = pd.read_csv('QVI_data.csv', parse_dates=['DATE'])

# Convert the date to a monthly period for aggregation
df['MONTH'] = df['DATE'].dt.to_period('M')
```

```
df.head()
```

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

```
# Group data by STORE and MONTH and calculate three main metrics
monthly_metrics = df.groupby(['STORE_NBR', 'MONTH']).agg({
    'TOT_SALES': 'sum',                        # Total Sales Revenue
    'LYLTY_CARD_NBR': pd.Series.nunique,       # Unique customers
    'TXN_ID': 'count'                          # Number of transactions
}).reset_index()

# Rename for clarity
monthly_metrics.rename(columns={
    'TOT_SALES': 'total_sales',
    'LYLTY_CARD_NBR': 'unique_customers',
    'TXN_ID': 'total_transactions'
}, inplace=True)

# Calculate transactions per customer
monthly_metrics['transactions_per_customer'] = monthly_metrics['total_transactions'] / monthly_metrics['unique_customers']
```

```
monthly_metrics.head()
```

| | STORE_NBR | MONTH | total_sales | unique_customers | total_transactions | transactions_per_customer |
|---|---|---|---|---|---|---|
| **0** | 1 | 2018-07 | 206.9 | 49 | 52 | 1.061224 |
| **1** | 1 | 2018-08 | 176.1 | 42 | 43 | 1.023810 |
| **2** | 1 | 2018-09 | 278.8 | 59 | 62 | 1.050847 |
| **3** | 1 | 2018-10 | 188.1 | 44 | 45 | 1.022727 |
| **4** | 1 | 2018-11 | 192.6 | 46 | 47 | 1.021739 |

Next steps: Generate code with monthly_metrics | View recommended plots | New interactive sheet

```
from scipy.stats import pearsonr
import numpy as np
```

```
def calculate_similarity(trial_store, candidate_store, metric='total_sales'):
    # Filter pre-trial data (before Feb 2019)
    pre_trial = monthly_metrics[monthly_metrics['MONTH'] < '2019-02']

    # Get data for trial and candidate stores
    trial_data = pre_trial[pre_trial['STORE_NBR'] == trial_store].sort_values('MONTH')
    candidate_data = pre_trial[pre_trial['STORE_NBR'] == candidate_store].sort_values('MONTH')

    # Ensure lengths match before comparing
    if len(trial_data) != len(candidate_data):
        return np.nan

    # Calculate Pearson correlation
    corr, _ = pearsonr(trial_data[metric], candidate_data[metric])
    return corr
```

```
trial_stores = [77, 86, 88]
all_stores = monthly_metrics['STORE_NBR'].unique()
control_matches = {}

for trial in trial_stores:
    similarities = []
    for store in all_stores:
        if store == trial:
            continue
        score = calculate_similarity(trial, store)
        similarities.append((store, score))

    # Choose the store with the highest correlation score
    best_match = max(similarities, key=lambda x: x[1])
    control_matches[trial] = best_match[0]

print("Control matches for trial stores:", control_matches)
```

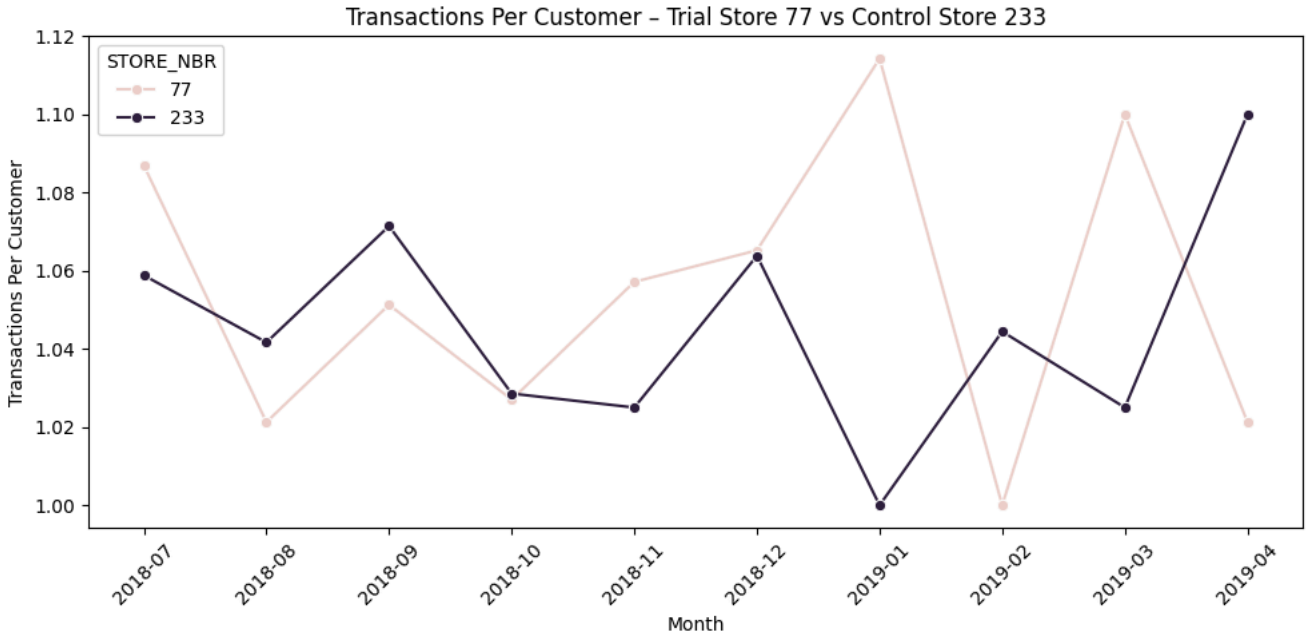
```
Control matches for trial stores: {77: np.int64(233), 86: np.int64(155), 88: np.int64(159)}
```

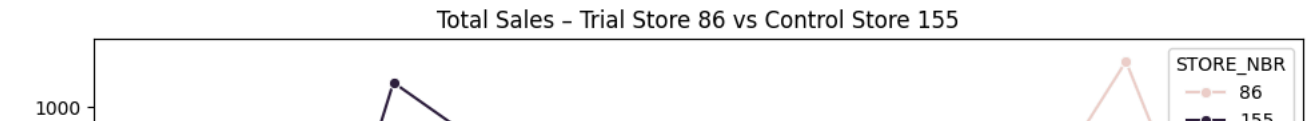
```
def plot_comparison(trial_store, control_store, metric):
    subset = monthly_metrics[monthly_metrics['STORE_NBR'].isin([trial_store, control_store])]
    subset = subset[subset['MONTH'] <= '2019-04']

    # 🔧 Convert 'MONTH' from Period to string for plotting
    subset['MONTH'] = subset['MONTH'].astype(str)

    # Plot
    plt.figure(figsize=(10, 5))
    sns.lineplot(data=subset, x='MONTH', y=metric, hue='STORE_NBR', marker='o')
    plt.title(f'{metric.replace("_", " ").title()} – Trial Store {trial_store} vs Control Store {control_store}')
    plt.xlabel('Month')
    plt.ylabel(metric.replace("_", " ").title())
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

```
# Plot for all 3 metrics and all 3 trial stores
for trial in trial_stores:
    control = control_matches[trial]
    plot_comparison(trial, control, 'total_sales')
    plot_comparison(trial, control, 'unique_customers')
    plot_comparison(trial, control, 'transactions_per_customer')
```


Total Sales – Trial Store 77 vs Control Store 233
Total Sales
Month
STORE_NBR
77
233
280
260
240
220
200
180
160
2018-07
2018-08
2018-09
2018-10
2018-11
2018-12
2019-01
2019-02
2019-03
2019-04


Unique Customers – Trial Store 77 vs Control Store 233
Unique Customers
Month
STORE_NBR
77
233
50
45
40
35
30
2018-07
2018-08
2018-09
2018-10
2018-11
2018-12
2019-01
2019-02
2019-03
2019-04


Transactions Per Customer – Trial Store 77 vs Control Store 233
Transactions Per Customer
Month
STORE_NBR
77
233
1.12
1.10
1.08
1.06
1.04
1.02
1.00
2018-07
2018-08
2018-09
2018-10
2018-11
2018-12
2019-01
2019-02
2019-03
2019-04


Total Sales – Trial Store 86 vs Control Store 155
STORE_NBR
86
1000