Performance trial analysis in stores 77, 86 and 88.

```python
import pandas as pd
behaviour = pd.read_csv('behaviour.csv')
behaviour.head()
```

index	LYLTY_CARD_NBR	LIFESTAGE
0	1000	YOUNG SINGLES/COUPLES
1	1002	YOUNG SINGLES/COUPLES
2	1003	YOUNG FAMILIES
3	1004	OLDER SINGLES/COUPLES
4	1005	MIDAGE SINGLES/COUPLES

Rows: 5

Expand

```python
# Change all column names in the transaction dataframe to lowercase
behaviour.columns = behaviour.columns.str.lower()
# Rename 'premium_customer' to 'customer_segment' and 'lylty_card_nbr' to 'loyalty_card_number' for clarity
behaviour = behaviour.rename(columns={
    'premium_customer': 'customer_segment',
    'lylty_card_nbr': 'loyalty_card_number'
})
behaviour.head()
```

index	loyalty_card_number	lifestage
0	1000	YOUNG SINGLES/COUPLES
1	1002	YOUNG SINGLES/COUPLES
2	1003	YOUNG FAMILIES
3	1004	OLDER SINGLES/COUPLES
4	1005	MIDAGE SINGLES/COUPLES

Rows: 5

Expand

```python
def map_situation(lifestage):
    if "SINGLES/COUPLES" in lifestage:
        return "Singles/Couples"
    elif "FAMILIES" in lifestage:
        return "Families"
    elif "RETIREES" in lifestage:
        return "Retirees"
    else:
        return "Other"

def map_age_category(lifestage):
    if "YOUNG" in lifestage or "NEW FAMILIES" in lifestage:
        return "Young"
    elif "MIDAGE" in lifestage:
        return "Midage"
    elif "OLDER" in lifestage:
        return "Older"
    elif "RETIREES" in lifestage:
        return "Retirees"
    else:
        return "Other"

behaviour['age_category'] = behaviour['lifestage'].apply(map_age_category)
behaviour['household_type'] = behaviour['lifestage'].apply(map_situation)
behaviour.head()
```

index	loyalty_card_number	lifestage	customer_segment
0	1000	YOUNG SINGLES/COUPLES	Premium
1	1002	YOUNG SINGLES/COUPLES	Mainstream
2	1003	YOUNG FAMILIES	Budget
3	1004	OLDER SINGLES/COUPLES	Mainstream
4	1005	MIDAGE SINGLES/COUPLES	Mainstream

Rows: 5

Expand

```
behaviour.drop(columns=['lifestage']).to_csv('customers.csv', index=False)
```

```
# Read in the transaction data
transactions = pd.read_csv('transaction.csv')

# Read in the customers data (created in previous steps)
customers = pd.read_csv('customers.csv')

# Ensure the join column is lowercase in both dataframes
transactions.columns = transactions.columns.str.lower()
if 'lylty_card_nbr' in transactions.columns:
    transactions = transactions.rename(columns={'lylty_card_nbr': 'loyalty_card_number'})

# Perform a left join on 'loyalty_card_number'
merged_df = transactions.merge(customers, on='loyalty_card_number', how='left')

# Display the first few rows of the merged dataframe
merged_df.head()
```

	d...	store...	loyalty_card_nu...	transact...	product...	product
0	2018-10-17	1	1000	1	5	Natural Chip Company Sea Salt
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly Sour Cream & Onion
4	2018-08-18	2	2426	1038	108	Kettle Tortilla Chips Honey & Jalapeno C

Rows: 5 Expand

```
print(merged_df['date'].min(), merged_df['date'].max())
```

```
2018-07-01 2019-06-30
```

```
import pandas as pd

# Define pre-trial and trial periods
pre_period = ('2018-07-01', '2018-12-31')   # July-Dec 2018
trial_period = ('2019-01-01', '2019-06-30') # Jan-Jun 2019
```

```python
# Monthly performance all stores
filtered_df = merged_df.copy()

# Ensure date is datetime
filtered_df['date'] = pd.to_datetime(filtered_df['date'])

# Create year and month name
filtered_df['year'] = filtered_df['date'].dt.year
filtered_df['month'] = filtered_df['date'].dt.month_name()

# Monthly sales & customer summary
monthly_summary = (
    filtered_df.groupby(['year', 'month', 'store_number'])
    .agg(
        total_sales=('total_sales', 'sum'),
        total_customers=('loyalty_card_number', 'nunique')
    )
    .reset_index()
)

# Average transactions per customer
transactions_per_customer = (
    filtered_df.groupby(['year', 'month', 'store_number', 'loyalty_card_number'])['transaction_id']
    .nunique()
    .groupby(['year', 'month', 'store_number'])
    .mean()
    .reset_index(name='avg_transactions')
)

# Merge results
monthly_summary = monthly_summary.merge(
    transactions_per_customer,
    on=['year', 'month', 'store_number'],
    how='left'
)

# Add proper date column
monthly_summary['date'] = pd.to_datetime(
    monthly_summary['year'].astype(str) + '-' + monthly_summary['month'],
    format='%Y-%B'
)
```

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

def find_control_store(data, trial_store, pre_period, trial_period, min_customers=20):
    # Get trial store pre-period data
    trial_df = data[(data['store_number']==trial_store) &
                    (data['date'].between(pre_period[0], pre_period[1]))]

    scores = []
    for store in data['store_number'].unique():
        # Skip the trial store itself and other trial stores
        if store == trial_store or store in [77, 86, 88]:
            continue

        # Candidate control store data
        control_pre = data[(data['store_number']==store) &
                           (data['date'].between(pre_period[0], pre_period[1]))]
        control_trial = data[(data['store_number']==store) &
                             (data['date'].between(trial_period[0], trial_period[1]))]

        # Skip if no trial data or too few customers
        if control_trial.empty or control_trial['total_customers'].mean() < min_customers:
            continue

        # Merge pre-period data for correlation
        merged = pd.merge(trial_df, control_pre, on='date', suffixes=('_trial','_control'))

        # Need at least 2 overlapping months for correlation
        if len(merged) >= 2:
            corr, _ = pearsonr(merged['total_sales_trial'], merged['total_sales_control'])
            scores.append({'trial_store': trial_store,
                           'control_store': store,
                           'corr': corr})

    scores_df = pd.DataFrame(scores)
    if not scores_df.empty:
        return scores_df.sort_values('corr', ascending=False).head(5)  # top 5 candidates
    else:
        return None
```

```python
pre_period = ('2018-07-01','2018-12-31')   # baseline
trial_period = ('2019-01-01','2019-06-30') # experiment window

best_control_77 = find_control_store(monthly_summary, 77, pre_period, trial_period)
best_control_86 = find_control_store(monthly_summary, 86, pre_period, trial_period)
best_control_88 = find_control_store(monthly_summary, 88, pre_period, trial_period)

print(best_control_77)
print(best_control_86)
print(best_control_88)
```

```
     trial_store  control_store      corr
63            77             71  0.944303
55            77             63  0.932288
103           77            119  0.885916
199           77            233  0.869930
2             77              3  0.853302
     trial_store  control_store      corr
152           86            178  0.908254
19            86             22  0.886097
133           86            155  0.876103
119           86            138  0.850964
206           86            240  0.824678
     trial_store  control_store      corr
160           88            186  0.905930
78            88             91  0.824997
138           88            163  0.809914
206           88            240  0.771855
116           88            134  0.760243
```

```python
from scipy.stats import ttest_ind

def compare_trial_vs_control(data, trial_store, control_store, trial_period):
    trial_df = data[(data['store_number']==trial_store) &
                    (data['date'].between(trial_period[0], trial_period[1]))]
    control_df = data[(data['store_number']==control_store) &
                      (data['date'].between(trial_period[0], trial_period[1]))]

    # Test if total sales differ
    t_stat, p_val = ttest_ind(trial_df['total_sales'], control_df['total_sales'])

    # Drivers of change
    avg_customers_trial = trial_df['total_customers'].mean()
    avg_customers_control = control_df['total_customers'].mean()

    avg_txn_trial = trial_df['avg_transactions'].mean()
    avg_txn_control = control_df['avg_transactions'].mean()

    return {
    'trial_store_id': trial_store,                 # ID of the store where the experiment ran
    'control_store_id': control_store,             # ID of the matched control store
    't_test_statistic': t_stat,                    # Result of the t-test (difference in sales)
    'p_value': p_val,                              # Significance level of the test
    'avg_monthly_customers_trial': avg_customers_trial,   # Average monthly customers in trial store
    'avg_monthly_customers_control': avg_customers_control, # Average monthly customers in control store
    'avg_transactions_per_customer_trial': avg_txn_trial,   # Avg transactions per customer in trial store
    'avg_transactions_per_customer_control': avg_txn_control # Avg transactions per customer in control store
}


trial_period = ('2019-01-01','2019-06-30')

result_77 = compare_trial_vs_control(monthly_summary, 77, int(best_control_77.iloc[0]['control_store']), trial_period)
result_86 = compare_trial_vs_control(monthly_summary, 86, int(best_control_86.iloc[0]['control_store']), trial_period)
result_88 = compare_trial_vs_control(monthly_summary, 88, int(best_control_88.iloc[0]['control_store']), trial_period)
```

```python
import pandas as pd

# Display the results as nicely formatted tables, if they are DataFrames
display(result_77)
display(result_86)
display(result_88)
```

```
{'trial_store_id': 77,
 'control_store_id': 71,
 't_test_statistic': -21.127369029916963,
 'p_value': 1.2546971387541633e-09,
 'avg_monthly_customers_trial': 45.5,
 'avg_monthly_customers_control': 110.66666666666667,
 'avg_transactions_per_customer_trial': 1.0463557286857752,
 'avg_transactions_per_customer_control': 1.2673986079667847}
```

```
{'trial_store_id': 86,
 'control_store_id': 178,
 't_test_statistic': -0.6124294799702276,
 'p_value': 0.5539238605876803,
 'avg_monthly_customers_trial': 103.83333333333333,
 'avg_monthly_customers_control': 104.5,
 'avg_transactions_per_customer_trial': 1.252383703245043,
 'avg_transactions_per_customer_control': 1.2624692414852976}
```

```
{'trial_store_id': 88,
 'control_store_id': 186,
 't_test_statistic': 35.02388768062617,
 'p_value': 8.53680019147897e-12,
 'avg_monthly_customers_trial': 125.33333333333333,
 'avg_monthly_customers_control': 35.166666666666664,
 'avg_transactions_per_customer_trial': 1.2362874540382802,
 'avg_transactions_per_customer_control': 1.0218677323940482}
```

Insight

- If the p-value is very small (e.g. < 0.05), it means the trial store's performance was significantly different from its control during the experimen
- Non-significant p-values (>0.05) mean the trial store tracked its control closely
- The **t-test** show the difference in sales_

Store 77 vs Control 71

- **t-stat** = –21.13, **p-val** ≈ 1.25e-09 → **0.00000000125 extremely significant difference.**
- Trial store averaged 45.5 customers/month, control had 110.7.
- Transactions per customer were lower in the trial (1.05 vs 1.27). This suggests Store 77 underperformed relative to its control during the trial period.
- The underperformance is mainly due to fewer customers, with a smaller contribution from lower activity per customer.

Store 86 vs Control 178

- **t-stat = –0.61, p-val ≈ 0.55 → no significant difference.**
- Customers were almost identical (103.8 vs 104.5).
- Transactions per customer were also very close (1.25 vs 1.26).
- Store 86 behaved very similarly to its control — the trial didn't cause a measurable change.

Store 88 vs Control 186

- **t-stat = 35.02, p-val ≈ 8.5e-12 → 0.0000000000085 extremely significant difference.**
- Trial store averaged 125.3 customers/month, control only 35.2.
- Transactions per customer were higher in the trial (1.24 vs 1.02).
- Store 88 clearly outperformed its control — the trial had a strong positive effect.
- The strong outperformance is driven mostly by more customers, with a smaller boost from higher activity per customer.

Recommendations

Store 77 underperformed compared to its control:

- Investigate customer acquisition and engagement strategies

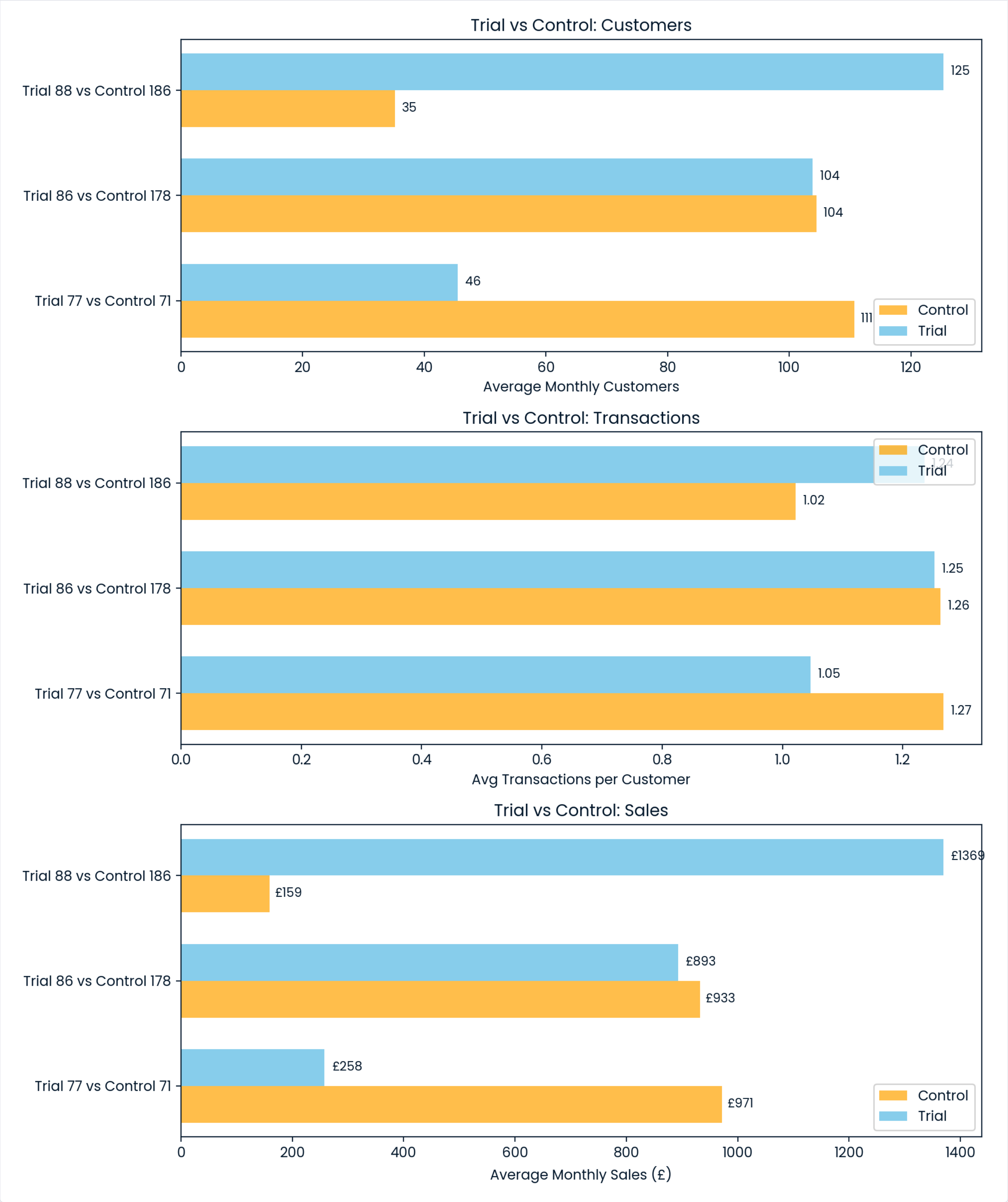
Store 86 tracked its control closely:

- no measurable trial effect.Treat this store as a neutral outcome

Store 88 strongly outperformed its control:

- Replicate the strategies used here — whatever was trialed worked well in driving customer growth and activity.

```python
import matplotlib.pyplot as plt
import numpy as np

# Collect results dynamically (these are your function outputs)
results = [result_77, result_86, result_88]

labels = [f"Trial {r['trial_store_id']} vs Control {r['control_store_id']}" for r in results]

trial_customers = [r['avg_monthly_customers_trial'] for r in results]
control_customers = [r['avg_monthly_customers_control'] for r in results]

trial_txn = [r['avg_transactions_per_customer_trial'] for r in results]
control_txn = [r['avg_transactions_per_customer_control'] for r in results]

# Example: also compute average monthly sales dynamically
trial_sales = [monthly_summary[(monthly_summary['store_number']==r['trial_store_id']) &
                               (monthly_summary['date'].between('2019-01-01','2019-06-30'))]['total_sales'].mean()
               for r in results]
control_sales = [monthly_summary[(monthly_summary['store_number']==r['control_store_id']) &
                                 (monthly_summary['date'].between('2019-01-01','2019-06-30'))]['total_sales'].mean()
                 for r in results]

fig, ax = plt.subplots(3, 1, figsize=(10, 12))
y_pos = np.arange(len(results))
bar_width = 0.35

def add_labels(bars, axis, offset=0.01, fmt="{:.2f}"):
    for bar in bars:
        width = bar.get_width()
        axis.text(width + offset*max([bar.get_width() for bar in bars]),
                  bar.get_y() + bar.get_height()/2,
                  fmt.format(width),
                  va='center', ha='left', fontsize=9)

# Customers
bars_control = ax[0].barh(y_pos - bar_width/2, control_customers, height=bar_width,
                          color='orange', alpha=0.7, label='Control')
bars_trial = ax[0].barh(y_pos + bar_width/2, trial_customers, height=bar_width,
                        color='skyblue', label='Trial')
ax[0].set_yticks(y_pos)
ax[0].set_yticklabels(labels)
ax[0].set_xlabel("Average Monthly Customers")
ax[0].set_title("Trial vs Control: Customers")
ax[0].legend()
add_labels(bars_control, ax[0], fmt="{:.0f}")
add_labels(bars_trial, ax[0], fmt="{:.0f}")

# Transactions per customer
bars_control_txn = ax[1].barh(y_pos - bar_width/2, control_txn, height=bar_width,
                              color='orange', alpha=0.7, label='Control')
bars_trial_txn = ax[1].barh(y_pos + bar_width/2, trial_txn, height=bar_width,
                            color='skyblue', label='Trial')
ax[1].set_yticks(y_pos)
ax[1].set_yticklabels(labels)
ax[1].set_xlabel("Avg Transactions per Customer")
ax[1].set_title("Trial vs Control: Transactions")
ax[1].legend()
add_labels(bars_control_txn, ax[1], fmt="{:.2f}")
add_labels(bars_trial_txn, ax[1], fmt="{:.2f}")

# Sales
bars_control_sales = ax[2].barh(y_pos - bar_width/2, control_sales, height=bar_width,
                                color='orange', alpha=0.7, label='Control')
bars_trial_sales = ax[2].barh(y_pos + bar_width/2, trial_sales, height=bar_width,
                              color='skyblue', label='Trial')
ax[2].set_yticks(y_pos)
ax[2].set_yticklabels(labels)
ax[2].set_xlabel("Average Monthly Sales (£)")
ax[2].set_title("Trial vs Control: Sales")
ax[2].legend()
add_labels(bars_control_sales, ax[2], fmt="£{:.0f}")
add_labels(bars_trial_sales, ax[2], fmt="£{:.0f}")

plt.tight_layout()
plt.show()
```


Trial vs Control: Customers
Trial 88 vs Control 186
Trial 86 vs Control 178
Trial 77 vs Control 71
125
35
104
104
46
111
Control
Trial
0
20
40
60
80
100
120
Average Monthly Customers
Trial vs Control: Transactions
Trial 88 vs Control 186
Trial 86 vs Control 178
Trial 77 vs Control 71
1.24
1.02
1.25
1.26
1.05
1.27
Control
Trial
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Avg Transactions per Customer
Trial vs Control: Sales
Trial 88 vs Control 186
Trial 86 vs Control 178
Trial 77 vs Control 71
£1369
£159
£893
£933
£258
£971
Control
Trial
0
200
400
600
800
1000
1200
1400
Average Monthly Sales (£)

```python
import pandas as pd

def find_top_products(data, trial_store, control_store, trial_period, product_col='product', top_n=5):
    """
    Compare product-level sales between a trial store and its control store
    during the trial period, and return the top N product gains and declines.
    """
    # Filter trial and control store data for the trial period
    trial_df = data[(data['store_number']==trial_store) &
                    (data['date'].between(trial_period[0], trial_period[1]))]
    control_df = data[(data['store_number']==control_store) &
                      (data['date'].between(trial_period[0], trial_period[1]))]

    # Aggregate sales by product
    trial_prod = trial_df.groupby(product_col)['total_sales'].sum().reset_index()
    control_prod = control_df.groupby(product_col)['total_sales'].sum().reset_index()

    # Merge trial vs control
    merged = pd.merge(trial_prod, control_prod, on=product_col, suffixes=('_trial','_control'))

    # Compute difference and percentage change
    merged['sales_diff'] = merged['total_sales_trial'] - merged['total_sales_control']
    merged['pct_change'] = (merged['sales_diff'] / merged['total_sales_control'].replace(0, pd.NA)) * 100
    merged['pct_change'] = merged['pct_change'].round(1)

    # Top gains and declines
    top_gains = merged.sort_values('sales_diff', ascending=False).head(top_n)
    top_declines = merged.sort_values('sales_diff', ascending=True).head(top_n)

    return top_gains, top_declines

# Example usage
trial_period = ('2019-01-01','2019-06-30')

top_gains_88, top_declines_88 = find_top_products(merged_df, 88, 186, trial_period, product_col='product', top_n=5)
top_gains_77, top_declines_77 = find_top_products(merged_df, 77, 71, trial_period, product_col='product', top_n=5)

top_gains_88
top_declines_88,
top_gains_77,
top_declines_77
```

	product	total_sales_tr...	total_sales_contr...	s...	p...
79	Smiths Crinkle Original	17.1	159.6	-142.5	-89.3
12	Doritos Corn Chips Cheese Supreme	22	140.8	-118.8	-84.4
94	Twisties Chicken	4.6	115	-110.4	-96
16	Doritos Corn Chips Supreme	13	117	-104	-88.9
36	Kettle Sweet Chilli And Sour Cream	16.2	97.2	-81	-83.3

Rows: 5 Expand

```python
import pandas as pd

def find_top_products(data, trial_store, control_store, trial_period, product_col='product', top_n=5):
    """
    Compare product-level sales between a trial store and its control store
    during the trial period, and return the top N product gains and declines.
    Declines are only products where trial sales < control sales (negative diff).
    """
    # Filter trial and control store data for the trial period
    trial_df = data[(data['store_number']==trial_store) &
                    (data['date'].between(trial_period[0], trial_period[1]))]
    control_df = data[(data['store_number']==control_store) &
                      (data['date'].between(trial_period[0], trial_period[1]))]

    # Aggregate sales by product
    trial_prod = trial_df.groupby(product_col)['total_sales'].sum().reset_index()
    control_prod = control_df.groupby(product_col)['total_sales'].sum().reset_index()

    # Merge trial vs control
    merged = pd.merge(trial_prod, control_prod, on=product_col, suffixes=('_trial','_control'))

    # Compute difference and percentage change
    merged['sales_diff'] = merged['total_sales_trial'] - merged['total_sales_control']
    merged['pct_change'] = (merged['sales_diff'] / merged['total_sales_control'].replace(0, pd.NA)) * 100

    # Top gains (positive differences only)
    top_gains = merged[merged['sales_diff'] > 0].sort_values('sales_diff', ascending=False).head(top_n)

    # Top declines (negative differences only)
    top_declines = merged[merged['sales_diff'] < 0].sort_values('sales_diff', ascending=True).head(top_n)

    return top_gains, top_declines

# Example usage
trial_period = ('2019-01-01','2019-06-30')

top_gains_88, top_declines_88 = find_top_products(merged_df, 88, 186, trial_period, product_col='product', top_n=5)
top_gains_77, top_declines_77 = find_top_products(merged_df, 77, 71, trial_period, product_col='product', top_n=5)
```

```python
def display_section(title, df, empty_message="No data", cmap="RdYlGn"):
    from IPython.display import display, HTML
    display(HTML(f"<h3>{title}</h3>"))
    if df is not None and not df.empty:
        styled = (
            df.head(5)
            .style
            .format({
                "total_sales_trial": "£{:,.2f}",
                "total_sales_control": "£{:,.2f}",
                "sales_diff": "£{:,.2f}",
                "pct_change": "{:+.1f}%"
            })
            .background_gradient(subset=["sales_diff", 'pct_change'], cmap=cmap)
        )
        display(styled)
    else:
        display(HTML(f"<i>{empty_message}</i>"))

# Usage
display_section("Store 88 vs 186 - Top 5 Gains", top_gains_88, cmap="Greens")
display_section("Store 88 vs 186 - Top 5 Declines", top_declines_88, empty_message="No declines", cmap="Reds")
display_section("Store 77 vs 71 - Top 5 Gains", top_gains_77, cmap="Greens")
display_section("Store 77 vs 71 - Top 5 Declines", top_declines_77, empty_message="No declines", cmap="Reds")
```

Store 88 Vs Control 186

The following products experienced remarkable sales increases during the trial period at Store 88 compared to its control store 186:

- **Smiths Crinkle Original**: £256.50 vs £11.40 ⟶ **+2150%**
- **Cobs Popd Sour Cream & Cheese Chips**: £220.40 vs £3.80 ⟶ **+5700%**
- **Kettle Sensations BBQ & Maple**: £230.00 vs £13.80 ⟶ **+1566.7%**
- **Cheezels Cheese**: £216.60 vs £11.40 ⟶ **+1800%**
- **Kettle Sea Salt & Vinegar**: £216.00 vs £10.80 ⟶ **+1900%** It suggests the trial store either gave these products better placement, stronger promotions, or tapped into local demand. The trial clearly worked. **No decline of sales in any product**

Store 77 vs Control 71 — Trial Underperformed

Very modest improvements:

- **Thins Potato Chips Hot & Spicy**: £29.70 vs £19.80 → +50%
- **Thins Chips Light & Tangy**: £16.50 vs £13.20 → +25% Significant drops:
- **Smiths Crinkle Origina**l: £17.10 vs £159.60 → –89.3%
- **Doritos Corn Chips Cheese Supreme**: £22.00 vs £140.80 → –84.4%
- **Twisties Chicken**: £4.60 vs £115.00 → –96%
- **Doritos Corn Chips Supreme**: £13.00 vs £117.00 → –88.9%
- **Kettle Sweet Chilli And Sour Cream**: £16.20 vs £97.20 → –83.3%

Recommendations Based on Sales Insights

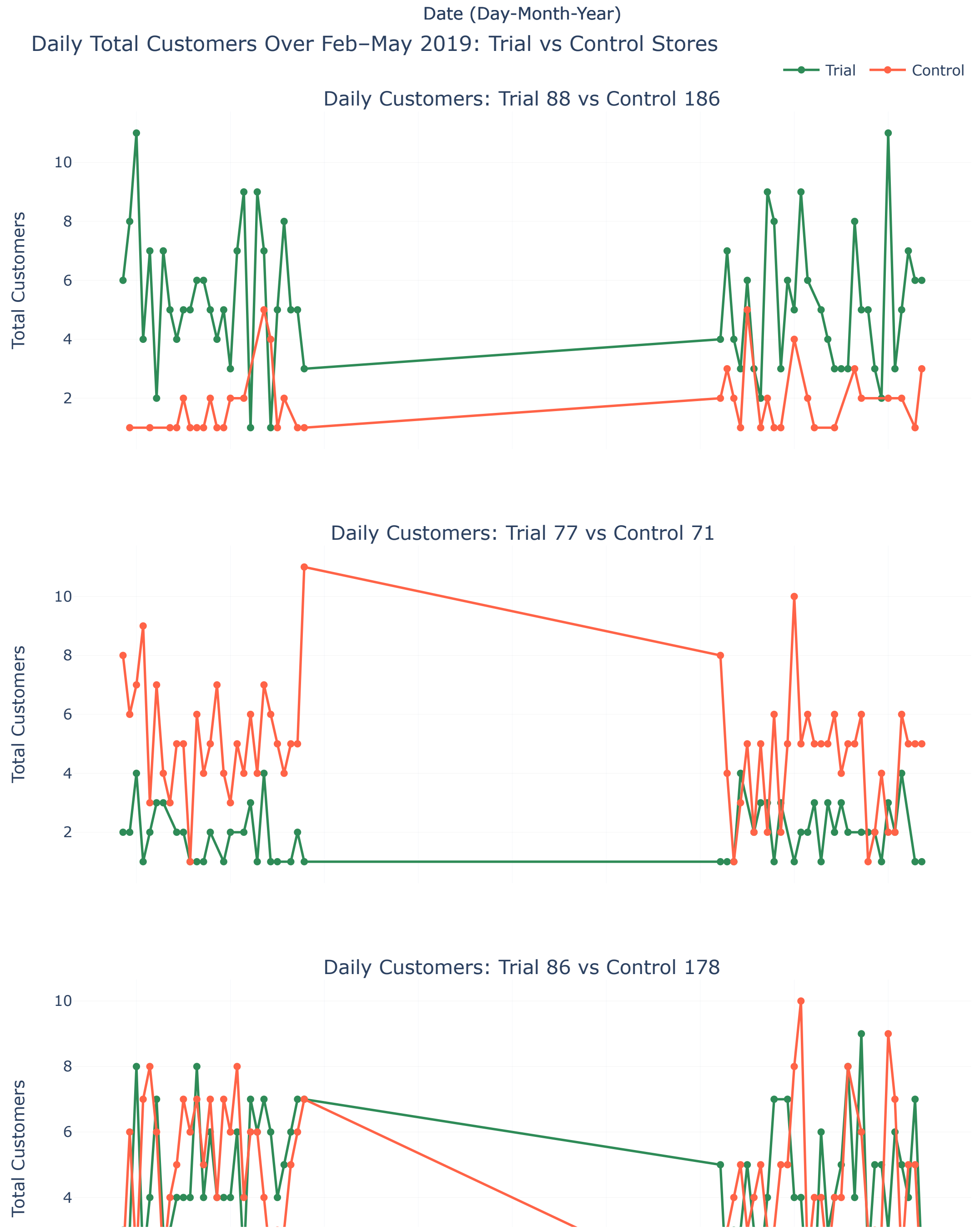
For Store 88

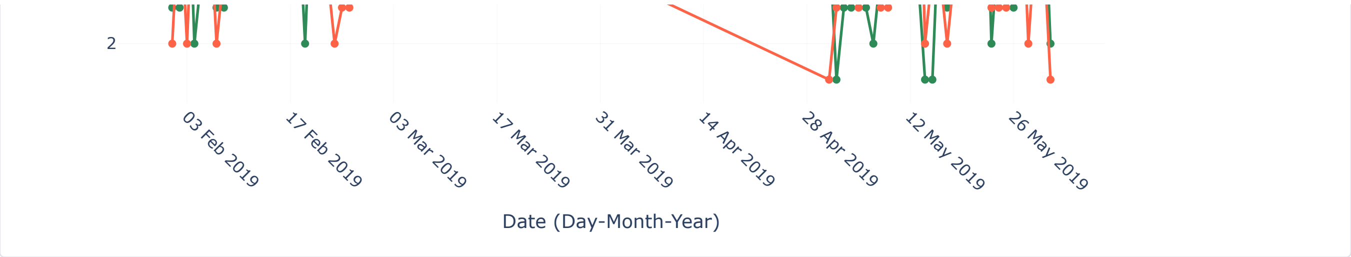
- **Continue or Expand Successful Strategies:** The trial at Store 88 led to dramatic sales increases for several products. Maintain or further invest in the tactics used (e.g., product placement, promotions, local targeting) for:
  - Smiths Crinkle Original
  - Cobs Popd Sour Cream & Cheese Chips
  - Kettle Sensations BBQ & Maple
  - Cheezels Cheese
  - Kettle Sea Salt & Vinegar
- **Replicate in Other Stores:** Consider rolling out the same strategies to similar stores to capture additional gains.
- **Monitor for Sustainability:** Track whether these sales lifts are sustained over time or if they were short-term spikes.

For Store 77

- **Reassess Trial Approach:** The trial underperformed, with most products experiencing significant sales declines. Investigate possible causes:
  - Was the execution of the trial different from Store 88?
  - Were there external factors (e.g., local competition, demographics) affecting results?
- **Targeted Interventions:** For products with modest gains (e.g., Thins Potato Chips Hot & Spicy, Thins Chips Light & Tangy), analyze what worked and see if those tactics can be applied to other products.
- **Address Declines:** For products with large declines, consider revising the assortment, pricing, or promotional strategy, or even removing underperforming SKUs if justified.

```
import pandas as pd
import plotly.graph_objs as go
from plotly.subplots import make_subplots

# --- Build daily_summary inside this cell ---
filtered_df = merged_df.copy()
filtered_df['date'] = pd.to_datetime(filtered_df['date'])

# Daily sales & customer summary
daily_summary = (
    filtered_df.groupby(['date', 'store_number'])
    .agg(
        total_sales=('total_sales', 'sum'),
        total_customers=('loyalty_card_number', 'nunique')
    )
    .reset_index()
)

# Average transactions per customer (daily)
transactions_per_customer = (
    filtered_df.groupby(['date', 'store_number', 'loyalty_card_number'])['transaction_id']
    .nunique()
    .groupby(['date', 'store_number'])
    .mean()
    .reset_index(name='avg_transactions')
)

# Merge results
daily_summary = daily_summary.merge(
    transactions_per_customer,
    on=['date', 'store_number'],
    how='left'
)

# --- Define trial/control pairs ---
results = [
    {'trial_store_id':88, 'control_store_id':186},
    {'trial_store_id':77, 'control_store_id':71},
    {'trial_store_id':86, 'control_store_id':178}
]

# --- Filter for Jan and Feb 2019 only ---
mask = daily_summary['date'].dt.to_period('M').isin(
    [pd.Period(year=2019, month=2, freq='M'), pd.Period(year=2019, month=5, freq='M')]
)
filtered_summary = daily_summary[mask]

# --- Plotting ---
fig = make_subplots(
    rows=len(results), cols=1,
    shared_xaxes=True,
    vertical_spacing=0.08,
    subplot_titles=[f"Daily Customers: Trial {r['trial_store_id']} vs Control {r['control_store_id']}" for r in results]
)

for i, r in enumerate(results):
    trial_df_plot = filtered_summary[
        filtered_summary['store_number'] == r['trial_store_id']
    ].sort_values('date')
    control_df_plot = filtered_summary[
        filtered_summary['store_number'] == r['control_store_id']
    ].sort_values('date')

    # Plot trial
    fig.add_trace(
        go.Scatter(
            x=trial_df_plot['date'],
            y=trial_df_plot['total_customers'],
            mode='lines+markers',
            name="Trial",
            marker=dict(color='seagreen'),
            legendgroup="Trial",
            showlegend=True if i == 0 else False
        ),
        row=i+1, col=1
    )
    # Plot control
    fig.add_trace(
        go.Scatter(
            x=control_df_plot['date'],
            y=control_df_plot['total_customers'],
            mode='lines+markers',
            name="Control",
            marker=dict(color='tomato'),
            legendgroup="Control",
            showlegend=True if i == 0 else False
        ),
        row=i+1, col=1
    )
    fig.update_yaxes(title_text="Total Customers", row=i+1, col=1,
                     showgrid=True, gridwidth=1, gridcolor='rgba(0,0,0,0.1)')

# Format x-axis as day-month-year
fig.update_xaxes(title_text="Date (Day-Month-Year)", tickformat="%d %b %Y", tickangle=45)

fig.update_layout(
    height=400 * len(results),
    width=900,
    title_text="Daily Total Customers Over Feb-May 2019: Trial vs Control Stores",
    legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1),
    template="plotly_white"
)

fig.show()
```




2
03 Feb 2019
17 Feb 2019
03 Mar 2019
17 Mar 2019
31 Mar 2019
14 Apr 2019
28 Apr 2019
12 May 2019
26 May 2019
Date (Day-Month-Year)