In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import ttest_ind, t
import matplotlib.dates as mdates
```

Importing Datasets

In [2]:
```
data = pd.read_csv("QVI_data.csv")
data
```

Out[2]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g
...	...	...	...	...	...	...
246734	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g
246735	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g
246736	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g
246737	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g
246738	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g

246739 rows × 12 columns

Data Wrangling

The client has selected store numbers 77, 86 and 88 as trial stores with a trial period of Feb 2019 to April 2019. The client also wants control stores to be established stores that are operational for the entire observation period.

We would want to match trial stores to control stores that are similar to the trial store

prior to the trial period of Feb 2019 in terms of:

Monthly overall sales revenue, Monthly number of customers, Monthly number of transactions per customer. To choose the control stores, we will create the metrics of interest and filter to stores that are present throughout the pre-trial period.

First, we want to add a column with the year/month of the transaction.

In [3]:
```
# Convert 'DATE' to datetime format
data['DATE'] = pd.to_datetime(data['DATE'])
# Add a column for year/month
data['YEARMONTH'] = data['DATE'].dt.strftime('%Y%m').astype('str')
data
```

Out[3]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g
...	...	...	...	...	...	...
246734	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g
246735	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g
246736	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g
246737	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g
246738	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g

246739 rows × 13 columns

Measures of Interests

For each store and month, calculate total sales, number of customers, transactions per customer, chips per customer and the average price per unit.

In [4]:
```
measure_overtime = data.groupby(['STORE_NBR', 'YEARMONTH']).agg(
    NUM_OF_TRANSACTIONS=pd.NamedAgg(column='STORE_NBR', aggfunc='count'),
```

```
    TOTAL_SALES=pd.NamedAgg(column='TOT_SALES', aggfunc='sum'),
    NUM_OF_CUSTOMERS=pd.NamedAgg(column='LYLTY_CARD_NBR', aggfunc='nunique')
    TOTAL_QTY=pd.NamedAgg(column='PROD_QTY', aggfunc='sum')).reset_index()

# Calculate additional metrics
measure_overtime['TRANSACTIONS_PER_CUSTOMER'] = (measure_overtime['NUM_OF_TR
measure_overtime['CHIPS_PER_CUSTOMER'] = (measure_overtime['TOTAL_QTY'] / me
measure_overtime['AVG_PRICE_PER_UNIT'] = (measure_overtime['TOTAL_SALES'] /

# Drop unnecessary columns
measure_overtime.drop(['NUM_OF_TRANSACTIONS','TOTAL_QTY'], axis=1, inplace=T


measure_overtime = pd.DataFrame(measure_overtime)
measure_overtime
```

Out[4]:

	STORE_NBR	YEARMONTH	TOTAL_SALES	NUM_OF_CUSTOMERS	TRANSACTIO
0	1	201807	188.9	47	
1	1	201808	168.4	41	
2	1	201809	268.1	57	
3	1	201810	175.4	39	
4	1	201811	184.8	44	
...	...	...	...	...	
3160	272	201902	385.3	44	
3161	272	201903	421.9	48	
3162	272	201904	445.1	54	
3163	272	201905	314.6	34	
3164	272	201906	301.9	33	

3165 rows × 7 columns

In [65]:

```
# Group by 'STORE_NBR' and filter stores with exactly 12 entries (months)
stores_with_full_obs = measure_overtime.groupby('STORE_NBR').filter(lambda x

# Get the unique store numbers that have full observation periods
stores_with_full_obs = stores_with_full_obs['STORE_NBR'].unique()

# Filter to the pre-trial period (before February 2019)
pre_trial_measures = measure_overtime[(measure_overtime['YEARMONTH'] < '2019
                                       (measure_overtime['STORE_NBR'].isin(sto

pre_trial_measures = pd.DataFrame(pre_trial_measures)
pre_trial_measures
```

Out[65]:

	STORE_NBR	YEARMONTH	TOTAL_SALES	NUM_OF_CUSTOMERS	TRANSACTIO
0	1	201807	188.9	47	
1	1	201808	168.4	41	
2	1	201809	268.1	57	
3	1	201810	175.4	39	
4	1	201811	184.8	44	
...	...	...	...	...	
3155	272	201809	294.5	31	
3156	272	201810	405.1	41	
3157	272	201811	355.8	39	
3158	272	201812	363.1	43	
3159	272	201901	392.4	44	

1813 rows × 7 columns

In [66]:
```
pre_trial_measures.describe()
```

Out[66]:

	STORE_NBR	TOTAL_SALES	NUM_OF_CUSTOMERS	TRANSACTIONS_PER_CUST
count	1813.000000	1813.000000	1813.000000	1813.0
mean	136.737452	586.001462	67.583012	1.1
std	78.561140	363.068511	34.563206	0.1
min	1.000000	3.000000	1.000000	1.0
25%	68.000000	260.500000	39.000000	1.0
50%	137.000000	654.500000	78.000000	1.1
75%	204.000000	879.000000	98.000000	1.2
max	272.000000	1557.600000	144.000000	1.5

In [61]:
```
during_trial_measures = measure_overtime[(measure_overtime['YEARMONTH'] >= '
                                          (measure_overtime['STORE_NBR'].isin(std
during_trial_measures.describe()
```

Out[61]:

	STORE_NBR	TOTAL_SALES	NUM_OF_CUSTOMERS	TRANSACTIONS_PER_CUST
count	777.000000	777.000000	777.000000	777.0
mean	136.737452	573.225740	65.894466	1.1
std	78.590060	357.558914	33.867072	0.1
min	1.000000	3.000000	1.000000	1.0
25%	68.000000	242.100000	38.000000	1.0
50%	137.000000	639.400000	74.000000	1.1
75%	204.000000	866.400000	96.000000	1.2
max	272.000000	1467.000000	133.000000	1.4

In Pre-Trial period, the average sales per store per month is $586, along with an average of 1.14 transactions per customer and 3.81 per chip unit. During the trial period (from February to April, 2019), the average sales per store per month is $573, indicating substential impact on the stores' transaction values. Even though with just 3 months of trial, multiple Key Performance Indicators, such as number of customers, transaction per customer, chips per customer and average price per unit remains relatively close to the 7 months of data pre-trial.

Correlation & Magnitude Distance Calculation

In [6]:
```
def calculate_corr(input_table, metric_col, trial_store):
    trial_data = input_table[input_table['STORE_NBR'] == trial_store]

    results = []

     # Loop through all stores in the dataset (excluding the trial store)
    for control_store in input_table['STORE_NBR'].unique():
        if control_store != trial_store:
            # Filter control store data
            control_data = input_table[input_table['STORE_NBR'] == control_s

            # Merge trial and control store data on YEARMONTH
            merged_data = pd.merge(trial_data, control_data, on="YEARMONTH",

            # Calculate the correlation for the selected metric
            correlation = merged_data[[f'{metric_col}_TRIAL', f'{metric_col}

            # Append the result (trial store, control store, and correlation
            results.append([trial_store, control_store, correlation])

    correlation_df = pd.DataFrame(results, columns=['STORE_NBR_TRIAL', 'STORE
```

```
    return correlation_df
```

In [7]:
```
def calculate_magnitude_distance(input_table, metric_col, trial_store):
    # Filter trial store data
    trial_data = input_table[input_table['STORE_NBR'] == trial_store]

    distance_vals = []

    # Loop through all stores in the dataset (excluding the trial store)
    for control_store in input_table['STORE_NBR'].unique():
        if control_store != trial_store:
            # Filter control store data
            control_data = input_table[input_table['STORE_NBR'] == control_s

            # Merge the trial and control store data on YEARMONTH
            merged_data = pd.merge(trial_data, control_data, on='YEARMONTH',

            # Calculate the absolute difference (magnitude distance)
            magnitude_distance = abs(merged_data[f'{metric_col}_TRIAL'] - me

            # Append the result (trial store, control store, YEARMONTH, and
            for year_month, dist in zip(merged_data['YEARMONTH'], magnitude_
                distance_vals.append([trial_store, control_store, year_month

    # Create a DataFrame for magnitude distances
    distance_df = pd.DataFrame(distance_vals, columns=['STORE_NBR_TRIAL', 'S

    # Standardize the magnitude distance (min-max normalization)
    min_max_dist = distance_df.groupby(['STORE_NBR_TRIAL', 'YEARMONTH'])['MA
    distance_df = pd.merge(distance_df, min_max_dist, on=['STORE_NBR_TRIAL',

    # Apply min-max normalization
    distance_df['magnitudeMeasure'] = 1 - (distance_df['MAGNITUDE_DISTANCE']

    # Calculate the mean of the standardized magnitude measure by trial and
    final_dist_df = distance_df.groupby(['STORE_NBR_TRIAL', 'STORE_NBR_CONTR
    final_dist_df.rename(columns={'magnitudeMeasure': 'MAGNITUDE'}, inplace=

    return final_dist_df
```

Helper Functions

In [8]:
```
def calculate_scores_for_trial(data, trial_store, metric):

    correlation = calculate_corr(data, metric, trial_store)
    magnitude = calculate_magnitude_distance(data, metric, trial_store)

    correlation.sort_values(by='CORRELATION', ascending=False, inplace=True)
    magnitude.sort_values(by='MAGNITUDE', ascending=False, inplace=True)

    # Combine correlation and magnitude for score calculation
    score = pd.concat([correlation, magnitude['MAGNITUDE']], axis=1)
```

```
    # Dynamically name the weighted score based on the metric
    weighted_score_column_name = f'WEIGHTED_SCORE_{metric.upper().split('_')

    score[weighted_score_column_name] = 0.5 * score['CORRELATION'] + 0.5 * s

    return score
```

In [9]:
```
def match_and_calculate_final_score(trial_store, score_sales, score_customer

    weighted_score_sales_name = "WEIGHTED_SCORE_SALES"
    weighted_score_customers_name = "WEIGHTED_SCORE_CUSTOMERS"
    # Ensure the correct column names are being accessed
    score_sales.set_index(['STORE_NBR_TRIAL', 'STORE_NBR_CONTROL'], inplace=
    score_customers.set_index(['STORE_NBR_TRIAL', 'STORE_NBR_CONTROL'], inpl

    # Concatenate the weighted scores from sales and customers
    score_control = pd.concat([score_sales[weighted_score_sales_name], score

    # Calculate the final control store score
    score_control['FINAL_CONTROL_SCORE'] = score_control[weighted_score_sale

    return score_control
```

In [10]:
```
def assign_store_types(measure_overtime_sales, trial_store, control_store):

    measure_overtime_sales['STORE_TYPE'] = measure_overtime_sales['STORE_NBR
        lambda x: 'Trial' if x == trial_store else ('Control' if x in contro
    return measure_overtime_sales
```

In [11]:
```
def plot_metric_by_month(past_data, metric):

    y_label = 'Total Sales' if metric == 'TOTAL_SALES' else 'Total Customers
    title = f"{y_label} by Month"

    # Plotting the data
    plt.figure(figsize=(14, 6), dpi=200)
    plt.title(title)
    plt.xlabel("Month of Operation")
    plt.ylabel(y_label)


    ax = sns.barplot(x='TRANSACTION_MONTH', y=metric, hue='STORE_TYPE', data

    plt.show()
```

In [12]:
```
def apply_scaling_factor(data, trial_value, control_value, control_column, m

    # Calculate the scaling factor
    scaling_factor = trial_value / control_value

    scaled_data = data[data['STORE_NBR'].isin(control_store)].copy()
```

```
    # Apply the scaling factor to the control column
    scaled_data[control_column] = (scaled_data[metric_column] * scaling_fact

    return scaled_data
```

In [13]:
```
def calculate_percentage_difference(scaled_control_data, trial_store, metric

    # Define the column names for control and trial data dynamically based o
    control_column_name = f'CONTROL_{metric.upper().split("_")[-1]}_SUM'
    trial_column_name = f'TRIAL_{metric.upper().split("_")[-1]}'

    # Extract the trial store's data
    trial_data = measure_overtime_data[measure_overtime_data['STORE_NBR'] ==

    # Merge the scaled control data with the trial data
    percentage_difference = pd.merge(scaled_control_data[['YEARMONTH', contr

    percentage_difference.rename(columns={metric: trial_column_name}, inplac

    # Calculate percentage difference
    percentage_difference['PERCENTAGE_DIFFERENCE'] = abs(percentage_differer

    return percentage_difference
```

In [14]:
```
def perform_hypothesis_test(percentage_difference, alpha=0.05):

    # Calculate STD in pre-trial period
    std_dev = percentage_difference[percentage_difference['YEARMONTH'] < '20
    df = 7
    t_critical = stats.t.ppf(0.95, df)
    print('H0: trial - pre_trial = 0 | Ha: trial - pre_trial =/= 0')
    print(f"Degree of Freedom: {df} | alpha = 0.05 | std_dev = {std_dev}")
    print(f"T-Critical Value: {t_critical}")

    percentage_difference['tValue'] = (percentage_difference['PERCENTAGE_DIF

    percentage_diff_trial_period = percentage_difference[(percentage_differe

    for (index, row) in percentage_diff_trial_period.iterrows():
        if row.tValue > t_critical:
            print(f"{row.YEARMONTH}: Reject H0. There is sufficient evidence
        else:
            print(f"{row.YEARMONTH}: Fail to Reject H0.")
    print(percentage_diff_trial_period)
```

In [15]:
```
def create_trial_assessment(measure_overtime_data, std_dev, metric='TOTAL_SA

    # Filter and prepare data for trial and control stores
    past_data = measure_overtime_data[measure_overtime_data['STORE_TYPE'].is

    past_data['TRANSACTION_MONTH'] = pd.to_datetime(past_data['YEARMONTH'].a
```

```
    # Create trial assessment for all stores (Trial + Control)
    trial_assessment = past_data[['STORE_TYPE', 'TRANSACTION_MONTH', 'YEARMO

    # Calculate Control store 95th percentile (metric * (1 + std_dev * 2))
    past_controls_95 = trial_assessment[trial_assessment['STORE_TYPE'] == 'C
    past_controls_95[metric] = past_controls_95[metric] * (1 + std_dev * 2)
    past_controls_95['STORE_TYPE'] = f'Control 95th % confidence interval'

    # Calculate Control store 5th percentile (metric * (1 - std_dev * 2))
    past_controls_5 = trial_assessment[trial_assessment['STORE_TYPE'] == 'Co
    past_controls_5[metric] = past_controls_5[metric] * (1 - std_dev * 2)
    past_controls_5['STORE_TYPE'] = f'Control 5th % confidence interval'

    # Combine the data
    trial_assessment = pd.concat([trial_assessment, past_controls_95, past_c

    trial_assessment.set_index('TRANSACTION_MONTH', inplace=True)

    return trial_assessment
```

In [16]:

```
def plot_data(trial_assessment, metric, highlight_period_start='201902', hig
              y_label=None, title=None, ylim=None):
    plt.figure(figsize=(12, 7))

    # Plot the data with a lineplot
    sns.lineplot(data=trial_assessment, x='TRANSACTION_MONTH', y=metric, hue

    # Highlight the trial period range using axvspan
    plt.axvspan(pd.to_datetime(f'{highlight_period_start[:4]}-{highlight_per
                pd.to_datetime(f'{highlight_period_end[:4]}-{highlight_perio
                color='orange', alpha=0.3, label=f'Trial Period ({highlight_

    # Labeling
    plt.xlabel('MONTH OF OPERATION', fontsize=14)
    plt.ylabel(y_label if y_label else metric.replace('_', ' ').title(), for
    plt.title(title if title else f'{metric.replace("_", " ").title()} BY MO

    # Set the x-axis to display months properly
    plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
    plt.xticks(rotation=45, fontsize=12)

    # If ylim is not passed, calculate dynamic limits based on the metric
    if ylim is None:
        min_value = trial_assessment[metric].min()
        max_value = trial_assessment[metric].max()
        margin = (max_value - min_value) * 0.8
        plt.ylim(0, max_value + margin)  # Increase margin
    else:
        plt.ylim(ylim)  # Use provided ylim range

    # Add a legend and grid
    plt.legend(title='Store Type', loc='upper left')
```

```
    plt.grid(True)

    # Tighten layout and show plot
    plt.tight_layout()
    plt.show()
```

Trial Store 77

In [17]:
```
trial_store = 77
```

In [18]:
```
# Calculate scores for each metric (Sales and Customers)
score_sales = calculate_scores_for_trial(pre_trial_measures, trial_store, 'T
score_customers = calculate_scores_for_trial(pre_trial_measures, trial_store

# Calculate final control store score
score_control = match_and_calculate_final_score(trial_store, score_sales, sc
score_control

# Get the best matching control store based on the final score
control_store_row = score_control.loc[score_control.groupby('STORE_NBR_TRIAL
control_store_row
```

Out[18]:

		WEIGHTED_SCORE_SALES	WEIGHTED_SC
STORE_NBR_TRIAL	**STORE_NBR_CONTROL**		
77	**233**	0.980368	

For Trial Store 77, the best matched Control Store is 233

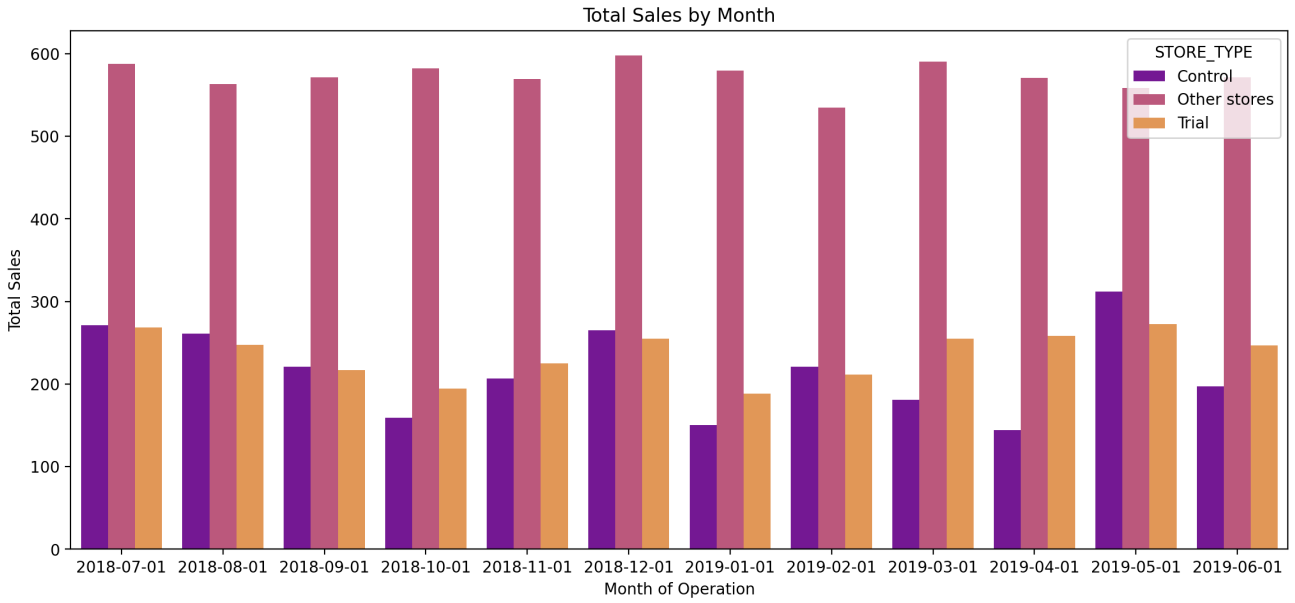
Sales Data Analysis

In [19]:
```
control_store = control_store_row.index.get_level_values('STORE_NBR_CONTROL'
# Assign store types ('Trial', 'Control', 'Other') to the measure_overtime_s
measure_overtime_sales = assign_store_types(measure_overtime.copy(), trial_s

# Group and calculate the mean of 'TOTAL_SALES' for each store type and mont
past_sales = measure_overtime_sales.groupby(['YEARMONTH', 'STORE_TYPE'])['TO

# Convert 'YEARMONTH' to a datetime object
past_sales['TRANSACTION_MONTH'] = pd.to_datetime(past_sales['YEARMONTH'].ast

# Plot total sales by month
plot_metric_by_month(past_sales, 'TOTAL_SALES')
```



In [20]:
```
# Apply scaling factors for sales
trial_sales = pre_trial_measures[(pre_trial_measures['STORE_NBR'] == trial_s
control_sales = pre_trial_measures[(pre_trial_measures['STORE_NBR'].isin(cor
scaled_control_sales = apply_scaling_factor(measure_overtime_sales, trial_sa
scaled_control_sales
```

Out[20]:

	STORE_NBR	YEARMONTH	TOTAL_SALES	NUM_OF_CUSTOMERS	TRANSACTIO
2695	233	201807	271.2	47	
2696	233	201808	260.7	44	
2697	233	201809	220.9	40	
2698	233	201810	159.3	32	
2699	233	201811	206.5	39	
2700	233	201812	265.4	43	
2701	233	201901	150.5	31	
2702	233	201902	220.7	42	
2703	233	201903	180.6	35	
2704	233	201904	144.2	27	
2705	233	201905	312.1	54	
2706	233	201906	197.0	34	

In [21]:
```
# Calculate percentage difference
percentage_difference_sales = calculate_percentage_difference(scaled_control
percentage_difference_sales
```

Out[21]:

	YEARMONTH	CONTROL_SALES_SUM	TRIAL_SALES	PERCENTAGE_DIFFERENCE
0	201807	282.0	268.4	0.048227
1	201808	271.1	247.5	0.087053
2	201809	229.7	216.8	0.056160
3	201810	165.6	194.3	0.173309
4	201811	214.7	224.9	0.047508
5	201812	276.0	255.2	0.075362
6	201901	156.5	188.4	0.203834
7	201902	229.5	211.6	0.077996
8	201903	187.8	255.1	0.358360
9	201904	149.9	258.1	0.721815
10	201905	324.5	272.3	0.160863
11	201906	204.8	246.6	0.204102

Hypothesis Testing

In [22]:
```
perform_hypothesis_test(percentage_difference_sales)
```

```
H0: trial - pre_trial = 0 | Ha: trial - pre_trial =/= 0
Degree of Freedom: 7 | alpha = 0.05 | std_dev = 0.0636103887674503
T-Critical Value: 1.894578605061305
201902: Fail to Reject H0.
201903: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
201904: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
  YEARMONTH  CONTROL_SALES_SUM  TRIAL_SALES  PERCENTAGE_DIFFERENCE     tValu
e
7    201902              229.5        211.6               0.077996   1.22614
6
8    201903              187.8        255.1               0.358360   5.63367
0
9    201904              149.9        258.1               0.721815  11.34743
2
```

Trial Assessment

In [23]:
```
std_dev = percentage_difference_sales[percentage_difference_sales['YEARMONTH
trial_assessment = create_trial_assessment(measure_overtime_sales, std_dev,
trial_assessment
```

```
/var/folders/7m/wfwdc8s14pgch1s3hgyyg6080000gn/T/ipykernel_95955/778200519.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  past_data['TRANSACTION_MONTH'] = pd.to_datetime(past_data['YEARMONTH'].astype(str), format='%Y%m')
```

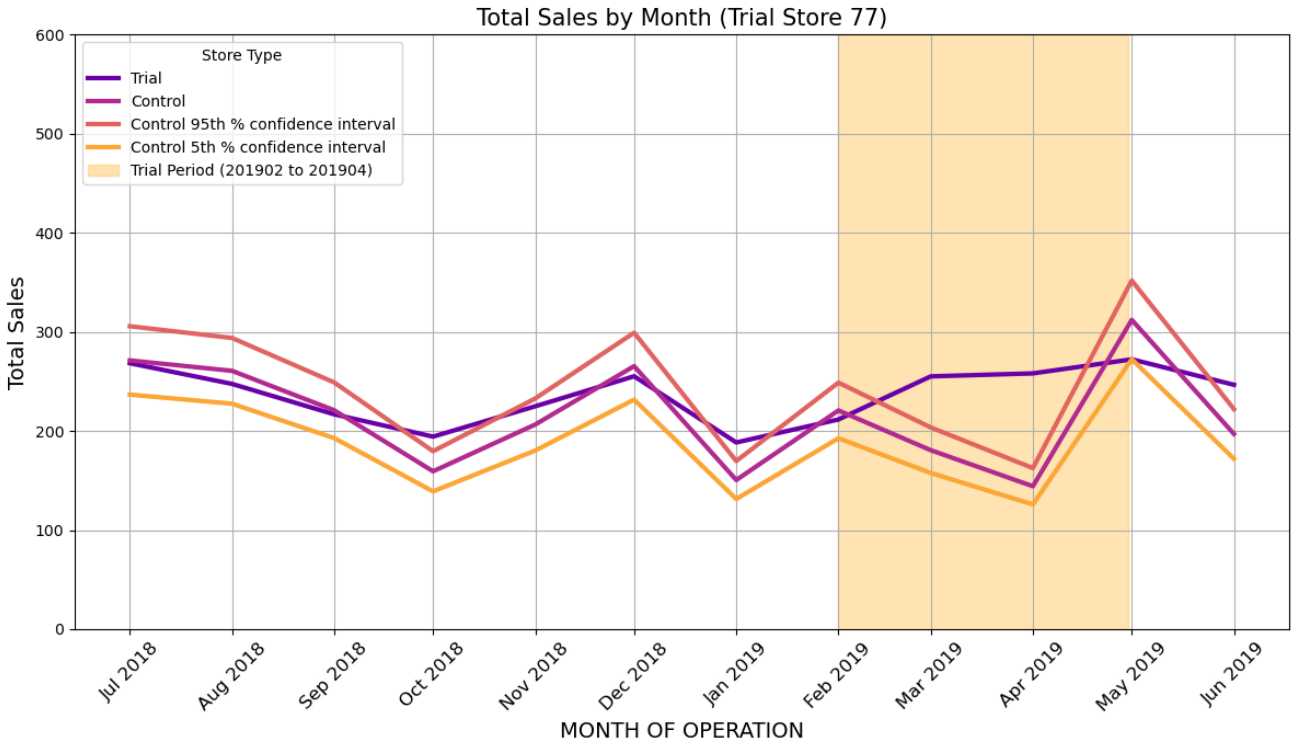
Out[23]:

TRANSACTION_MONTH	STORE_TYPE	YEARMONTH	TOTAL_SALES
2018-07-01	Trial	201807	268.400000
2018-08-01	Trial	201808	247.500000
2018-09-01	Trial	201809	216.800000
2018-10-01	Trial	201810	194.300000
2018-11-01	Trial	201811	224.900000
2018-12-01	Trial	201812	255.200000
2019-01-01	Trial	201901	188.400000
2019-02-01	Trial	201902	211.600000
2019-03-01	Trial	201903	255.100000
2019-04-01	Trial	201904	258.100000
2019-05-01	Trial	201905	272.300000
2019-06-01	Trial	201906	246.600000
2018-07-01	Control	201807	271.200000
2018-08-01	Control	201808	260.700000
2018-09-01	Control	201809	220.900000
2018-10-01	Control	201810	159.300000
2018-11-01	Control	201811	206.500000
2018-12-01	Control	201812	265.400000
2019-01-01	Control	201901	150.500000
2019-02-01	Control	201902	220.700000
2019-03-01	Control	201903	180.600000
2019-04-01	Control	201904	144.200000
2019-05-01	Control	201905	312.100000
2019-06-01	Control	201906	197.000000

2018-07-01	Control 95th % confidence interval	201807	305.702275
2018-08-01	Control 95th % confidence interval	201808	293.866457
2018-09-01	Control 95th % confidence interval	201809	249.003070
2018-10-01	Control 95th % confidence interval	201810	179.566270
2018-11-01	Control 95th % confidence interval	201811	232.771091
2018-12-01	Control 95th % confidence interval	201812	299.164394
2019-01-01	Control 95th % confidence interval	201901	169.646727
2019-02-01	Control 95th % confidence interval	201902	248.777626
2019-03-01	Control 95th % confidence interval	201903	203.576072
2019-04-01	Control 95th % confidence interval	201904	162.545236
2019-05-01	Control 95th % confidence interval	201905	351.805605
2019-06-01	Control 95th % confidence interval	201906	222.062493
2018-07-01	Control 5th % confidence interval	201807	236.697725
2018-08-01	Control 5th % confidence interval	201808	227.533543
2018-09-01	Control 5th % confidence interval	201809	192.796930
2018-10-01	Control 5th % confidence interval	201810	139.033730
2018-11-01	Control 5th % confidence interval	201811	180.228909
2018-12-01	Control 5th % confidence interval	201812	231.635606
2019-01-01	Control 5th % confidence interval	201901	131.353273
2019-02-01	Control 5th % confidence interval	201902	192.622374

	2019-03-01	Control 5th % confidence interval	201903	157.623928
	2019-04-01	Control 5th % confidence interval	201904	125.854764
	2019-05-01	Control 5th % confidence interval	201905	272.394395
	2019-06-01	Control 5th % confidence interval	201906	171.937507

In [24]:
```
# Plot the sales data for the trial store and control stores
plot_data(
    trial_assessment,
    metric='TOTAL_SALES',
    y_label='Total Sales',
    title=f'Total Sales by Month (Trial Store {trial_store})',
    ylim=(0,600)
    )
```



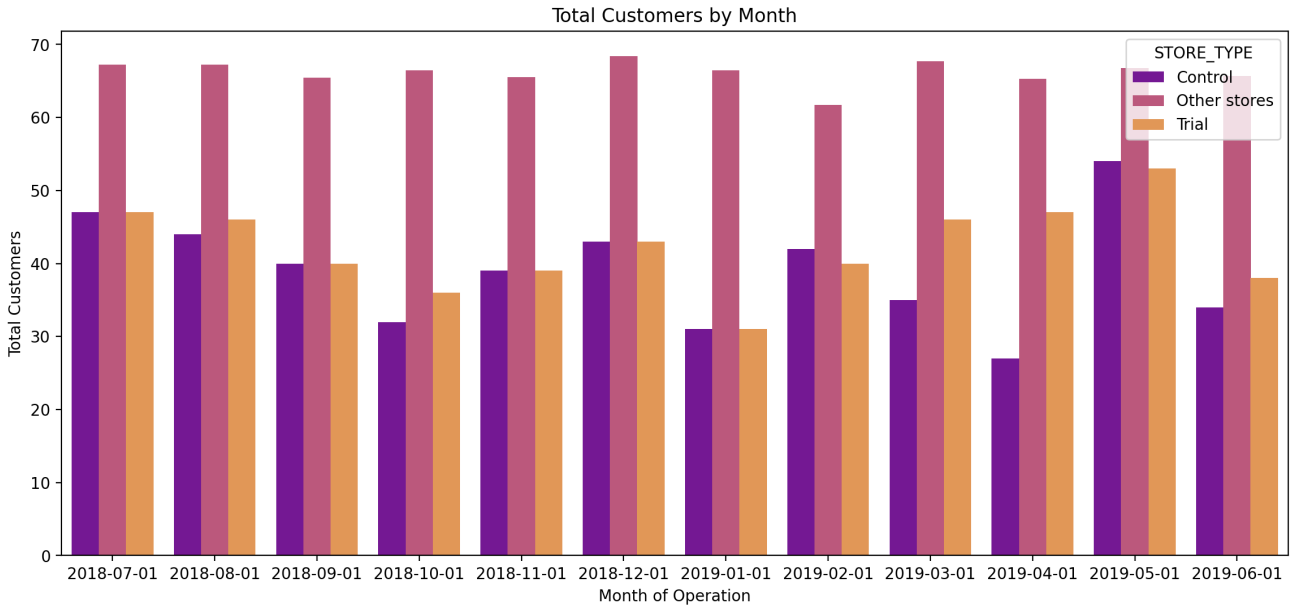
Customers Data Analysis

In [25]:
```
control_store = control_store_row.index.get_level_values('STORE_NBR_CONTROL'
# Assign store types ('Trial', 'Control', 'Other') to the measure_overtime_s
measure_overtime_customers = assign_store_types(measure_overtime.copy(), tri

# Group and calculate the mean of 'TOTAL_SALES' for each store type and mont
past_customers = measure_overtime_sales.groupby(['YEARMONTH', 'STORE_TYPE'])
```

```
# Convert 'YEARMONTH' to a datetime object
past_customers['TRANSACTION_MONTH'] = pd.to_datetime(past_customers['YEARMON

# Plot total sales by month
plot_metric_by_month(past_customers, 'NUM_OF_CUSTOMERS')
```



In [26]:
```
# Apply scaling factors for customers
trial_customers = pre_trial_measures[(pre_trial_measures['STORE_NBR'] == tri
control_customers = pre_trial_measures[(pre_trial_measures['STORE_NBR'].isir
scaled_control_customers = apply_scaling_factor(measure_overtime, trial_cust
scaled_control_customers
```

Out[26]:

	STORE_NBR	YEARMONTH	TOTAL_SALES	NUM_OF_CUSTOMERS	TRANSACTIO
2695	233	201807	271.2	47	
2696	233	201808	260.7	44	
2697	233	201809	220.9	40	
2698	233	201810	159.3	32	
2699	233	201811	206.5	39	
2700	233	201812	265.4	43	
2701	233	201901	150.5	31	
2702	233	201902	220.7	42	
2703	233	201903	180.6	35	
2704	233	201904	144.2	27	
2705	233	201905	312.1	54	
2706	233	201906	197.0	34	

In [27]:
```
# Calculate percentage difference
percentage_difference_customers = calculate_percentage_difference(scaled_cor
percentage_difference_customers
```

Out[27]:

	YEARMONTH	CONTROL_CUSTOMERS_SUM	TRIAL_CUSTOMERS	PERCENTAGE_DIF
0	201807	48.0	47	
1	201808	45.0	46	
2	201809	40.9	40	
3	201810	32.7	36	
4	201811	39.8	39	
5	201812	43.9	43	
6	201901	31.7	31	
7	201902	42.9	40	
8	201903	35.8	46	
9	201904	27.6	47	
10	201905	55.2	53	
11	201906	34.7	38	

Hypothesis Testing

In [28]:
```
perform_hypothesis_test(percentage_difference_customers)
```

```
H0: trial - pre_trial = 0 | Ha: trial - pre_trial =/= 0
Degree of Freedom: 7 | alpha = 0.05 | std_dev = 0.030107855697612342
T-Critical Value: 1.894578605061305
201902: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
201903: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
201904: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
  YEARMONTH  CONTROL_CUSTOMERS_SUM  TRIAL_CUSTOMERS  PERCENTAGE_DIFFERENCE
\
7    201902                  42.9               40               0.067599
8    201903                  35.8               46               0.284916
9    201904                  27.6               47               0.702899

      tValue
7   2.245230
8   9.463185
9  23.346018
```

Trial Assessment

In [29]:
```
# Trial Assessment:
std_dev = percentage_difference_customers[percentage_difference_customers['Y
trial_assessment = create_trial_assessment(measure_overtime_customers, std_d
trial_assessment
```

```
/var/folders/7m/wfwdc8s14pgch1s3hgyyg6080000gn/T/ipykernel_95955/778200519.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  past_data['TRANSACTION_MONTH'] = pd.to_datetime(past_data['YEARMONTH'].ast
ype(str), format='%Y%m')
```

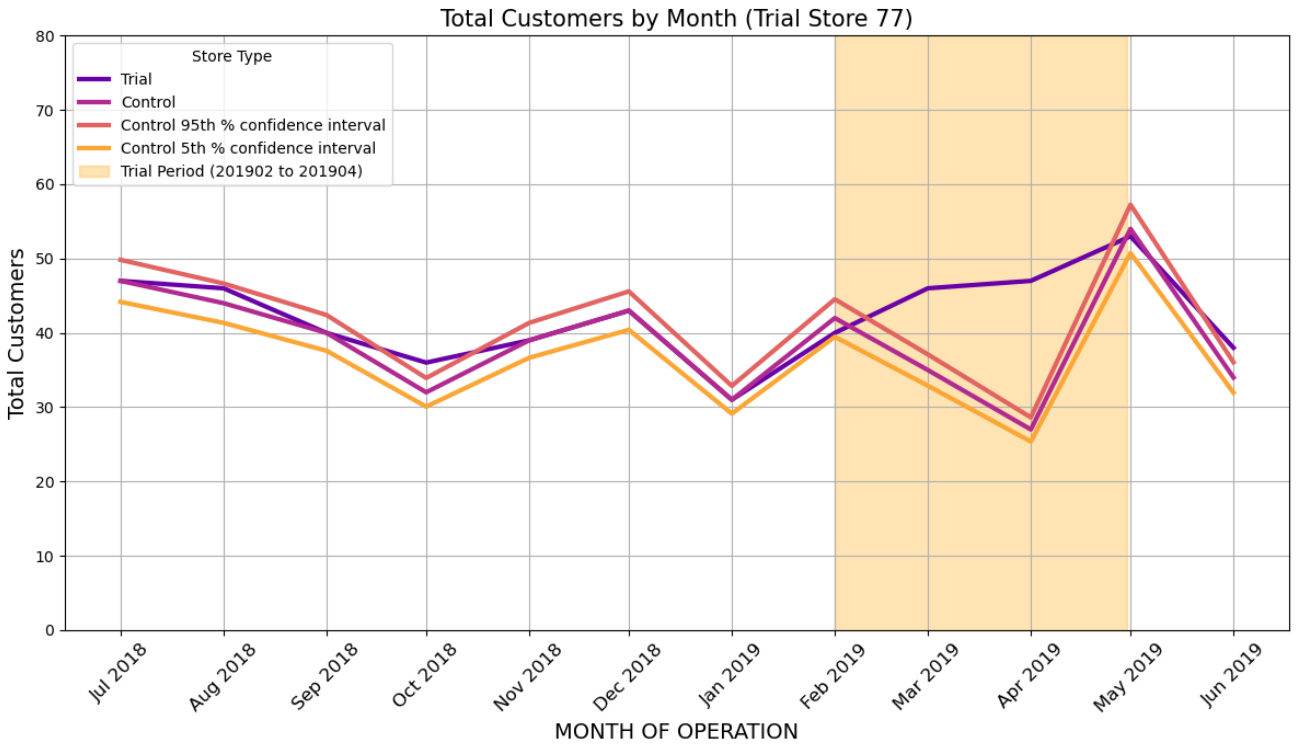
Out[29]:

TRANSACTION_MONTH	STORE_TYPE	YEARMONTH	NUM_OF_CUSTOMERS
2018-07-01	Trial	201807	47.000000
2018-08-01	Trial	201808	46.000000
2018-09-01	Trial	201809	40.000000
2018-10-01	Trial	201810	36.000000
2018-11-01	Trial	201811	39.000000
2018-12-01	Trial	201812	43.000000

2019-01-01	Trial	201901	31.000000
2019-02-01	Trial	201902	40.000000
2019-03-01	Trial	201903	46.000000
2019-04-01	Trial	201904	47.000000
2019-05-01	Trial	201905	53.000000
2019-06-01	Trial	201906	38.000000
2018-07-01	Control	201807	47.000000
2018-08-01	Control	201808	44.000000
2018-09-01	Control	201809	40.000000
2018-10-01	Control	201810	32.000000
2018-11-01	Control	201811	39.000000
2018-12-01	Control	201812	43.000000
2019-01-01	Control	201901	31.000000
2019-02-01	Control	201902	42.000000
2019-03-01	Control	201903	35.000000
2019-04-01	Control	201904	27.000000
2019-05-01	Control	201905	54.000000
2019-06-01	Control	201906	34.000000
2018-07-01	Control 95th % confidence interval	201807	49.830138
2018-08-01	Control 95th % confidence interval	201808	46.649491
2018-09-01	Control 95th % confidence interval	201809	42.408628
2018-10-01	Control 95th % confidence interval	201810	33.926903
2018-11-01	Control 95th % confidence interval	201811	41.348413
2018-12-01	Control 95th % confidence interval	201812	45.589276
2019-01-01	Control 95th % confidence interval	201901	32.866687
2019-02-01	Control 95th % confidence interval	201902	44.529060
	Control 95th %		

2019-03-01	confidence interval	201903	37.107550
2019-04-01	Control 95th % confidence interval	201904	28.625824
2019-05-01	Control 95th % confidence interval	201905	57.251648
2019-06-01	Control 95th % confidence interval	201906	36.047334
2018-07-01	Control 5th % confidence interval	201807	44.169862
2018-08-01	Control 5th % confidence interval	201808	41.350509
2018-09-01	Control 5th % confidence interval	201809	37.591372
2018-10-01	Control 5th % confidence interval	201810	30.073097
2018-11-01	Control 5th % confidence interval	201811	36.651587
2018-12-01	Control 5th % confidence interval	201812	40.410724
2019-01-01	Control 5th % confidence interval	201901	29.133313
2019-02-01	Control 5th % confidence interval	201902	39.470940
2019-03-01	Control 5th % confidence interval	201903	32.892450
2019-04-01	Control 5th % confidence interval	201904	25.374176
2019-05-01	Control 5th % confidence interval	201905	50.748352
2019-06-01	Control 5th % confidence interval	201906	31.952666

In [30]:
```
# Plot the data for NUM_OF_CUSTOMERS
plot_data(
        trial_assessment,
        metric='NUM_OF_CUSTOMERS',
        y_label='Total Customers',
        title=f'Total Customers by Month (Trial Store {trial_store})',
        ylim=(0,80)
    )
```



Trial Store 86

In [31]:
```
trial_store = 86
```

In [32]:
```
# Calculate scores for each metric (Sales and Customers)
score_sales = calculate_scores_for_trial(pre_trial_measures, trial_store, 'T
score_customers = calculate_scores_for_trial(pre_trial_measures, trial_store

# Calculate final control store score
score_control = match_and_calculate_final_score(trial_store, score_sales, sc
score_control

# Get the best matching control store based on the final score
control_store_row = score_control.loc[score_control.groupby('STORE_NBR_TRIAL
control_store_row
```

Out[32]:

		WEIGHTED_SCORE_SALES	WEIGHTED_SC(
STORE_NBR_TRIAL	**STORE_NBR_CONTROL**		
86	**155**	0.91645	

For Trial Store 86, the best matched Control Store is 155

Sales Data Analysis

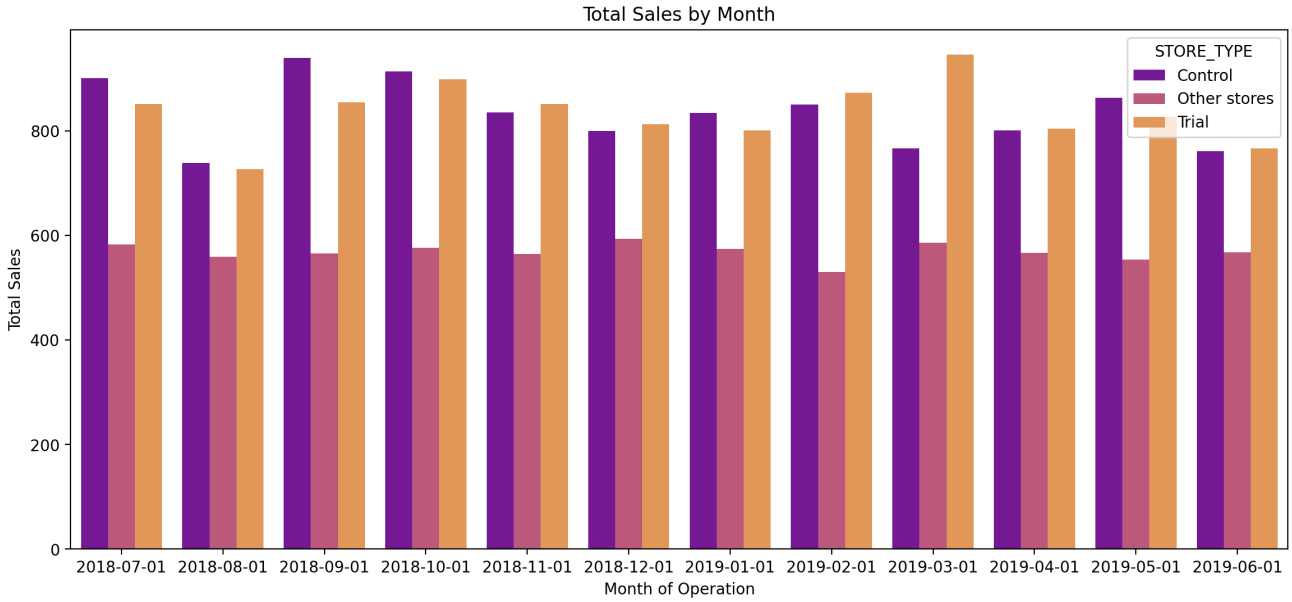
In [33]:
```
control_store = control_store_row.index.get_level_values('STORE_NBR_CONTROL'
# Assign store types ('Trial', 'Control', 'Other') to the measure_overtime_s
```

```
measure_overtime_sales = assign_store_types(measure_overtime.copy(), trial_s

# Group and calculate the mean of 'TOTAL_SALES' for each store type and mont
past_sales = measure_overtime_sales.groupby(['YEARMONTH', 'STORE_TYPE'])['TO

# Convert 'YEARMONTH' to a datetime object
past_sales['TRANSACTION_MONTH'] = pd.to_datetime(past_sales['YEARMONTH'].ast

# Plot total sales by month
plot_metric_by_month(past_sales, 'TOTAL_SALES')
```



In [34]:
```
# Apply scaling factors for sales
trial_sales = pre_trial_measures[(pre_trial_measures['STORE_NBR'] == trial_s
control_sales = pre_trial_measures[(pre_trial_measures['STORE_NBR'].isin(cor
scaled_control_sales = apply_scaling_factor(measure_overtime_sales, trial_sa
scaled_control_sales
```

Out[34]:

	STORE_NBR	YEARMONTH	TOTAL_SALES	NUM_OF_CUSTOMERS	TRANSACTIO
1792	155	201807	900.60	98	
1793	155	201808	738.70	88	
1794	155	201809	939.60	96	
1795	155	201810	914.00	105	
1796	155	201811	835.00	96	
1797	155	201812	799.80	91	
1798	155	201901	834.60	92	
1799	155	201902	850.80	92	
1800	155	201903	767.00	91	
1801	155	201904	800.40	93	
1802	155	201905	863.25	101	
1803	155	201906	760.80	87	

In [35]:
```
# Calculate percentage difference
percentage_difference_sales = calculate_percentage_difference(scaled_control
percentage_difference_sales
```

Out[35]:

	YEARMONTH	CONTROL_SALES_SUM	TRIAL_SALES	PERCENTAGE_DIFFERENCE
0	201807	875.4	851.00	0.027873
1	201808	718.1	726.85	0.012185
2	201809	913.3	855.00	0.063834
3	201810	888.5	898.80	0.011593
4	201811	811.7	851.20	0.048663
5	201812	777.4	812.20	0.044765
6	201901	811.3	800.60	0.013189
7	201902	827.0	872.80	0.055381
8	201903	745.6	945.40	0.267972
9	201904	778.0	804.00	0.033419
10	201905	839.1	826.90	0.014539
11	201906	739.5	766.00	0.035835

Hypothesis Testing

In [36]:
```
perform_hypothesis_test(percentage_difference_sales)
```

```
H0: trial - pre_trial = 0 | Ha: trial - pre_trial =/= 0
Degree of Freedom: 7 | alpha = 0.05 | std_dev = 0.020950396547484092
T-Critical Value: 1.894578605061305
201902: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
201903: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
201904: Fail to Reject H0.
  YEARMONTH  CONTROL_SALES_SUM  TRIAL_SALES  PERCENTAGE_DIFFERENCE     tValu
e
7    201902              827.0        872.8               0.055381   2.64342
9
8    201903              745.6        945.4               0.267972  12.79078
9
9    201904              778.0        804.0               0.033419   1.59515
0
```

Trial Assessment

In [37]:
```
std_dev = percentage_difference_sales[percentage_difference_sales['YEARMONTH
trial_assessment = create_trial_assessment(measure_overtime_sales, std_dev,
trial_assessment
```

```
/var/folders/7m/wfwdc8s14pgch1s3hgyyg6080000gn/T/ipykernel_95955/778200519.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  past_data['TRANSACTION_MONTH'] = pd.to_datetime(past_data['YEARMONTH'].ast
ype(str), format='%Y%m')
```

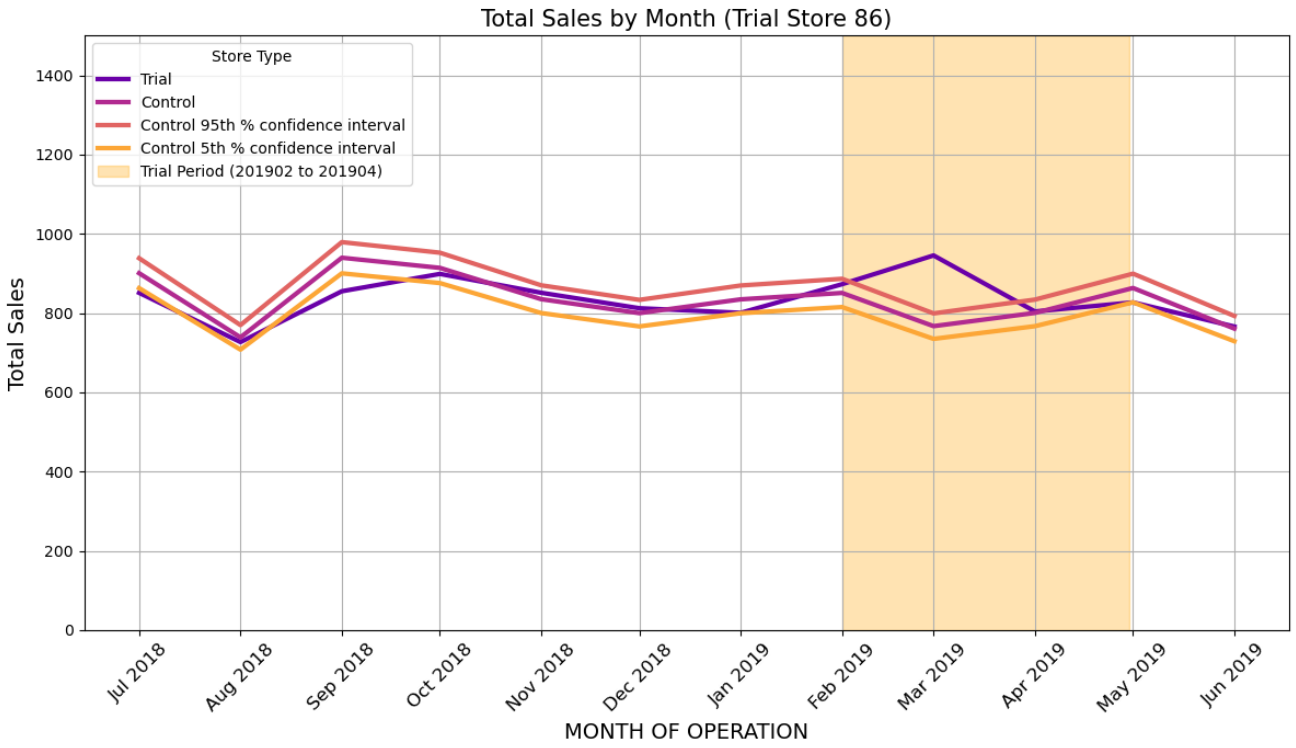
Out[37]:

TRANSACTION_MONTH	STORE_TYPE	YEARMONTH	TOTAL_SALES
2018-07-01	Trial	201807	851.000000
2018-08-01	Trial	201808	726.850000
2018-09-01	Trial	201809	855.000000
2018-10-01	Trial	201810	898.800000
2018-11-01	Trial	201811	851.200000
2018-12-01	Trial	201812	812.200000
2019-01-01	Trial	201901	800.600000
2019-02-01	Trial	201902	872.800000
2019-03-01	Trial	201903	945.400000

2019-04-01	Trial	201904	804.000000
2019-05-01	Trial	201905	826.900000
2019-06-01	Trial	201906	766.000000
2018-07-01	Control	201807	900.600000
2018-08-01	Control	201808	738.700000
2018-09-01	Control	201809	939.600000
2018-10-01	Control	201810	914.000000
2018-11-01	Control	201811	835.000000
2018-12-01	Control	201812	799.800000
2019-01-01	Control	201901	834.600000
2019-02-01	Control	201902	850.800000
2019-03-01	Control	201903	767.000000
2019-04-01	Control	201904	800.400000
2019-05-01	Control	201905	863.250000
2019-06-01	Control	201906	760.800000
2018-07-01	Control 95th % confidence interval	201807	938.335854
2018-08-01	Control 95th % confidence interval	201808	769.652116
2018-09-01	Control 95th % confidence interval	201809	978.969985
2018-10-01	Control 95th % confidence interval	201810	952.297325
2018-11-01	Control 95th % confidence interval	201811	869.987162
2018-12-01	Control 95th % confidence interval	201812	833.312254
2019-01-01	Control 95th % confidence interval	201901	869.570402
2019-02-01	Control 95th % confidence interval	201902	886.449195
2019-03-01	Control 95th % confidence interval	201903	799.137908
2019-04-01	Control 95th % confidence interval	201904	833.937395
	Control 95th % confidence		

2019-05-01	interval	201905	899.420860
2019-06-01	Control 95th % confidence interval	201906	792.678123
2018-07-01	Control 5th % confidence interval	201807	862.864146
2018-08-01	Control 5th % confidence interval	201808	707.747884
2018-09-01	Control 5th % confidence interval	201809	900.230015
2018-10-01	Control 5th % confidence interval	201810	875.702675
2018-11-01	Control 5th % confidence interval	201811	800.012838
2018-12-01	Control 5th % confidence interval	201812	766.287746
2019-01-01	Control 5th % confidence interval	201901	799.629598
2019-02-01	Control 5th % confidence interval	201902	815.150805
2019-03-01	Control 5th % confidence interval	201903	734.862092
2019-04-01	Control 5th % confidence interval	201904	766.862605
2019-05-01	Control 5th % confidence interval	201905	827.079140
2019-06-01	Control 5th % confidence interval	201906	728.921877

In [38]:
```
# Plot the sales data for the trial store and control stores
plot_data(
    trial_assessment,
    metric='TOTAL_SALES',
    y_label='Total Sales',
    title=f'Total Sales by Month (Trial Store {trial_store})',
    ylim=(0,1500)
    )
```



Despite of the significant increase in customer volume, the total sales for Trial Store 86 did not surpass the Control Store's 95th percentile in all 3 months.

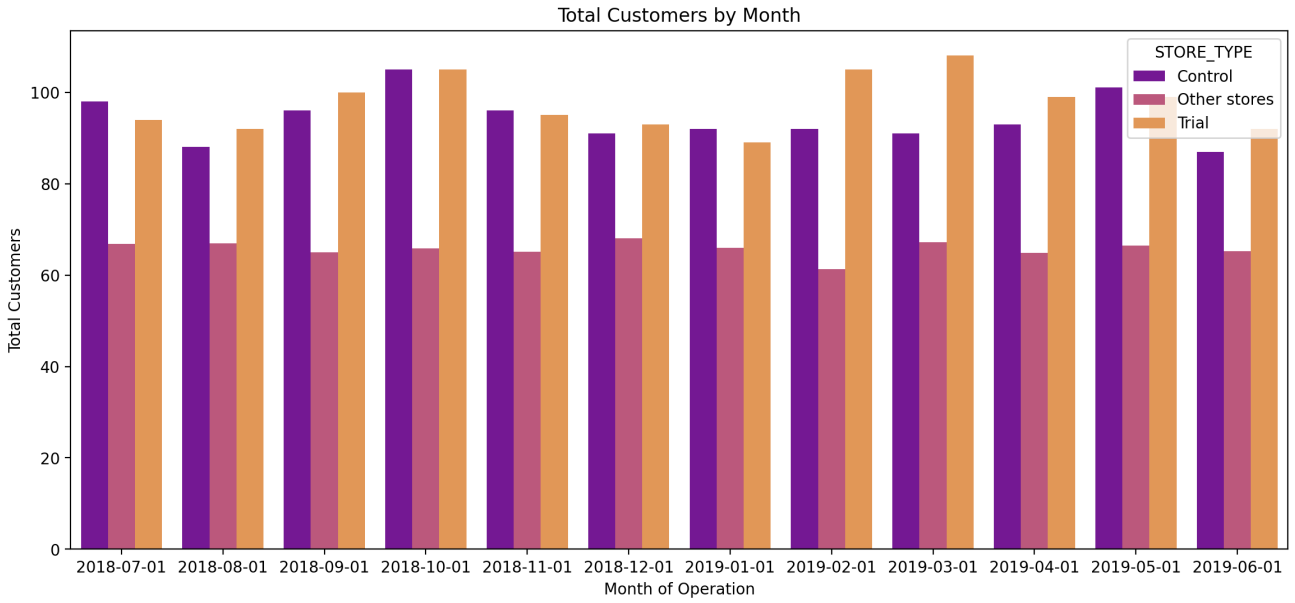
Customers Data Analysis

In [39]:
```
control_store = control_store_row.index.get_level_values('STORE_NBR_CONTROL'
# Assign store types ('Trial', 'Control', 'Other') to the measure_overtime_s
measure_overtime_customers = assign_store_types(measure_overtime.copy(), tri

# Group and calculate the mean of 'TOTAL_SALES' for each store type and mont
past_customers = measure_overtime_sales.groupby(['YEARMONTH', 'STORE_TYPE'])

# Convert 'YEARMONTH' to a datetime object
past_customers['TRANSACTION_MONTH'] = pd.to_datetime(past_customers['YEARMON

# Plot total sales by month
plot_metric_by_month(past_customers, 'NUM_OF_CUSTOMERS')
```



In [40]:
```
# Apply scaling factors for customers
trial_customers = pre_trial_measures[(pre_trial_measures['STORE_NBR'] == tri
control_customers = pre_trial_measures[(pre_trial_measures['STORE_NBR'].isir
scaled_control_customers = apply_scaling_factor(measure_overtime, trial_cust
scaled_control_customers
```

Out[40]:

	STORE_NBR	YEARMONTH	TOTAL_SALES	NUM_OF_CUSTOMERS	TRANSACTIO
1792	155	201807	900.60	98	
1793	155	201808	738.70	88	
1794	155	201809	939.60	96	
1795	155	201810	914.00	105	
1796	155	201811	835.00	96	
1797	155	201812	799.80	91	
1798	155	201901	834.60	92	
1799	155	201902	850.80	92	
1800	155	201903	767.00	91	
1801	155	201904	800.40	93	
1802	155	201905	863.25	101	
1803	155	201906	760.80	87	

In [41]:
```
# Calculate percentage difference
percentage_difference_customers = calculate_percentage_difference(scaled_cor
percentage_difference_customers
```

Out[41]:

	YEARMONTH	CONTROL_CUSTOMERS_SUM	TRIAL_CUSTOMERS	PERCENTAGE_DIF
0	201807	98.3	94	
1	201808	88.3	92	
2	201809	96.3	100	
3	201810	105.3	105	
4	201811	96.3	95	
5	201812	91.3	93	
6	201901	92.3	89	
7	201902	92.3	105	
8	201903	91.3	108	
9	201904	93.3	99	
10	201905	101.3	99	
11	201906	87.3	92	

Hypothesis Testing

In [42]:
```
perform_hypothesis_test(percentage_difference_customers)
```

```
H0: trial - pre_trial = 0 | Ha: trial - pre_trial =/= 0
Degree of Freedom: 7 | alpha = 0.05 | std_dev = 0.016023300886540606
T-Critical Value: 1.894578605061305
201902: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
201903: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
201904: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
  YEARMONTH  CONTROL_CUSTOMERS_SUM  TRIAL_CUSTOMERS  PERCENTAGE_DIFFERENCE
\
7    201902                  92.3              105               0.137595
8    201903                  91.3              108               0.182913
9    201904                  93.3               99               0.061093

      tValue
7   8.587169
8  11.415468
9   3.812775
```

Trial Assessment

In [43]:
```
# Trial Assessment:
std_dev = percentage_difference_customers[percentage_difference_customers['Y
```

```
trial_assessment = create_trial_assessment(measure_overtime_customers, std_d
trial_assessment
```

```
/var/folders/7m/wfwdc8s14pgch1s3hgyyg6080000gn/T/ipykernel_95955/778200519.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  past_data['TRANSACTION_MONTH'] = pd.to_datetime(past_data['YEARMONTH'].ast
ype(str), format='%Y%m')
```

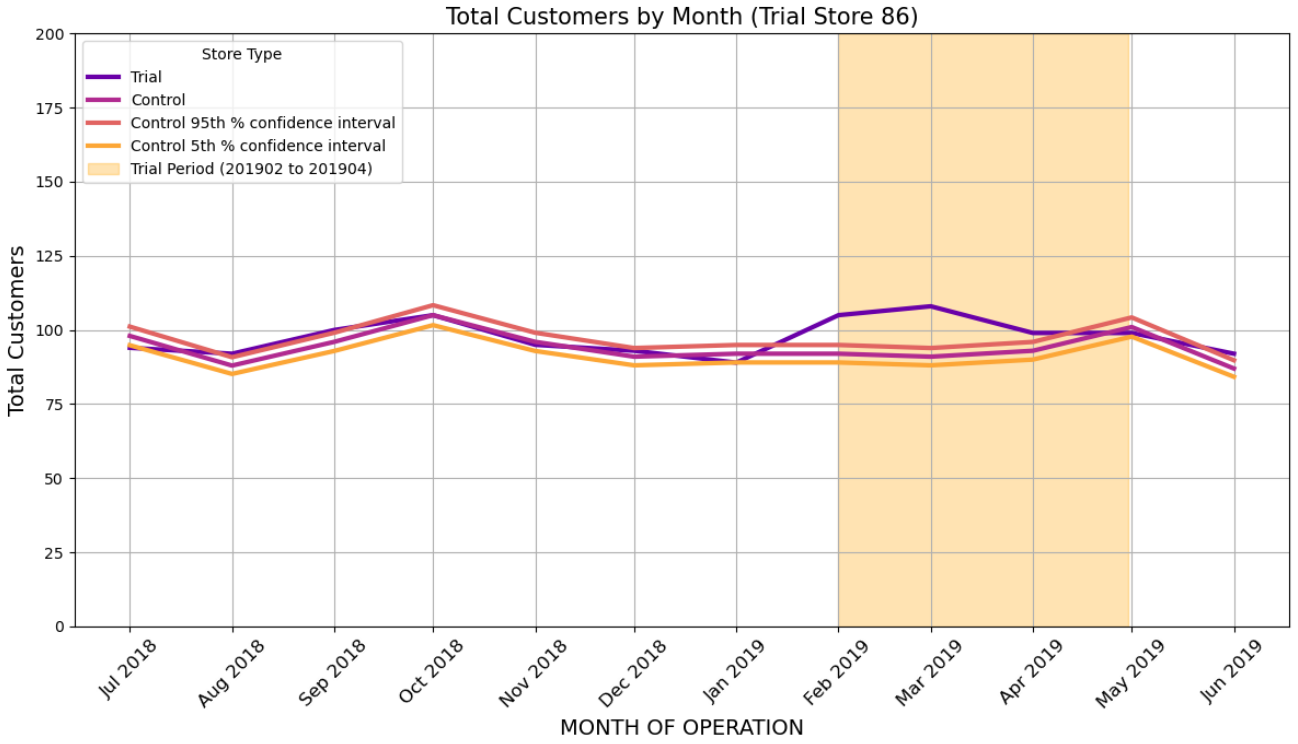
Out[43]:

TRANSACTION_MONTH	STORE_TYPE	YEARMONTH	NUM_OF_CUSTOMERS
2018-07-01	Trial	201807	94.000000
2018-08-01	Trial	201808	92.000000
2018-09-01	Trial	201809	100.000000
2018-10-01	Trial	201810	105.000000
2018-11-01	Trial	201811	95.000000
2018-12-01	Trial	201812	93.000000
2019-01-01	Trial	201901	89.000000
2019-02-01	Trial	201902	105.000000
2019-03-01	Trial	201903	108.000000
2019-04-01	Trial	201904	99.000000
2019-05-01	Trial	201905	99.000000
2019-06-01	Trial	201906	92.000000
2018-07-01	Control	201807	98.000000
2018-08-01	Control	201808	88.000000
2018-09-01	Control	201809	96.000000
2018-10-01	Control	201810	105.000000
2018-11-01	Control	201811	96.000000
2018-12-01	Control	201812	91.000000
2019-01-01	Control	201901	92.000000
2019-02-01	Control	201902	92.000000
2019-03-01	Control	201903	91.000000
2019-04-01	Control	201904	93.000000

2019-05-01	Control	201905	101.000000
2019-06-01	Control	201906	87.000000
2018-07-01	Control 95th % confidence interval	201807	101.140567
2018-08-01	Control 95th % confidence interval	201808	90.820101
2018-09-01	Control 95th % confidence interval	201809	99.076474
2018-10-01	Control 95th % confidence interval	201810	108.364893
2018-11-01	Control 95th % confidence interval	201811	99.076474
2018-12-01	Control 95th % confidence interval	201812	93.916241
2019-01-01	Control 95th % confidence interval	201901	94.948287
2019-02-01	Control 95th % confidence interval	201902	94.948287
2019-03-01	Control 95th % confidence interval	201903	93.916241
2019-04-01	Control 95th % confidence interval	201904	95.980334
2019-05-01	Control 95th % confidence interval	201905	104.236707
2019-06-01	Control 95th % confidence interval	201906	89.788054
2018-07-01	Control 5th % confidence interval	201807	94.859433
2018-08-01	Control 5th % confidence interval	201808	85.179899
2018-09-01	Control 5th % confidence interval	201809	92.923526
2018-10-01	Control 5th % confidence interval	201810	101.635107
2018-11-01	Control 5th % confidence interval	201811	92.923526
2018-12-01	Control 5th % confidence interval	201812	88.083759
2019-01-01	Control 5th % confidence interval	201901	89.051713

2019-02-01	Control 5th % confidence interval	201902	89.051713
2019-03-01	Control 5th % confidence interval	201903	88.083759
2019-04-01	Control 5th % confidence interval	201904	90.019666
2019-05-01	Control 5th % confidence interval	201905	97.763293
2019-06-01	Control 5th % confidence interval	201906	84.211946

In [44]:
```
# Plot the data for NUM_OF_CUSTOMERS
plot_data(
        trial_assessment,
        metric='NUM_OF_CUSTOMERS',
        y_label='Total Customers',
        title=f'Total Customers by Month (Trial Store {trial_store})',
        ylim=(0,200)
    )
```



It looks like the number of customers is significantly higher in all of the three months. This seems to suggest that the trial had a significant impact on increasing the number of customers in trial store 86 but as we saw, sales were not significantly higher. We should check with the Category Manager if there were special deals in the trial store that were may have resulted in lower prices, impacting the results.

Trial Store 88

In [45]:
```
trial_store = 88
```

In [46]:
```
# Calculate scores for each metric (Sales and Customers)
score_sales = calculate_scores_for_trial(pre_trial_measures, trial_store, 'T
score_customers = calculate_scores_for_trial(pre_trial_measures, trial_store

# Calculate final control store score
score_control = match_and_calculate_final_score(trial_store, score_sales, sc
score_control

# Get the best matching control store based on the final score
control_store_row = score_control.loc[score_control.groupby('STORE_NBR_TRIAL
control_store_row
```

Out[46]:

		WEIGHTED_SCORE_SALES	WEIGHTED_SC
STORE_NBR_TRIAL	**STORE_NBR_CONTROL**		
88	**237**	0.529452	

For Trial Store 88, the best matched Control Store is 237

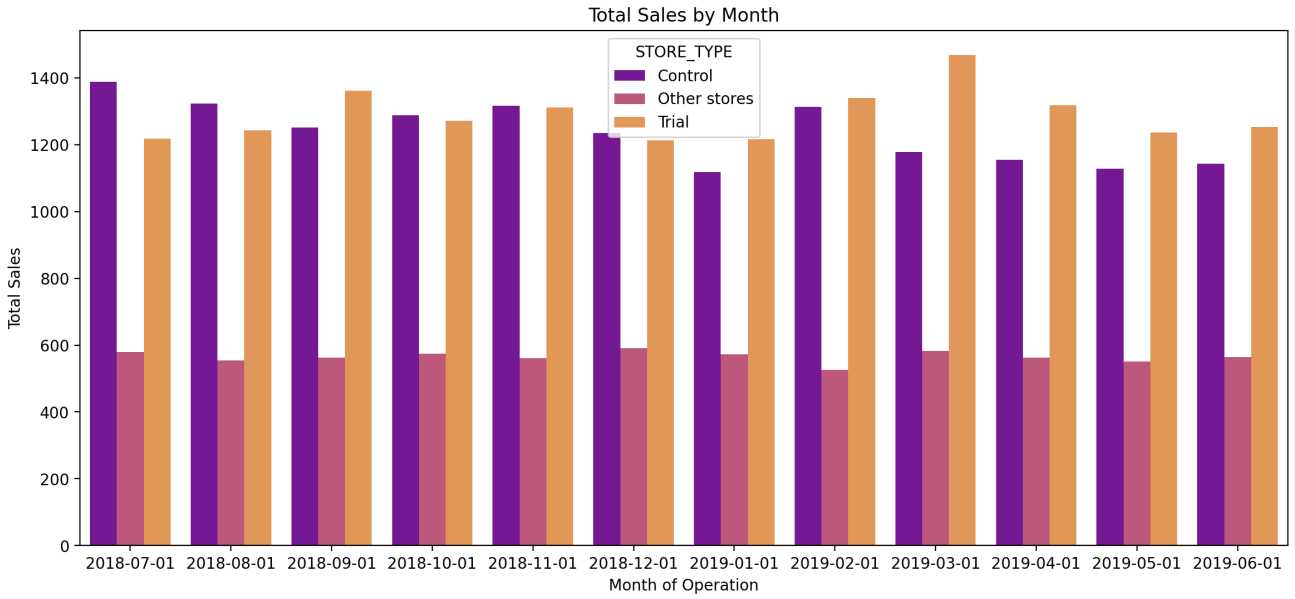
Sales Data Analysis

In [47]:
```
control_store = control_store_row.index.get_level_values('STORE_NBR_CONTROL'
# Assign store types ('Trial', 'Control', 'Other') to the measure_overtime_s
measure_overtime_sales = assign_store_types(measure_overtime.copy(), trial_s

# Group and calculate the mean of 'TOTAL_SALES' for each store type and mont
past_sales = measure_overtime_sales.groupby(['YEARMONTH', 'STORE_TYPE'])['TO

# Convert 'YEARMONTH' to a datetime object
past_sales['TRANSACTION_MONTH'] = pd.to_datetime(past_sales['YEARMONTH'].ast

# Plot total sales by month
plot_metric_by_month(past_sales, 'TOTAL_SALES')
```



In [48]:
```
# Apply scaling factors for sales so control and trial stores are comparable
trial_sales = pre_trial_measures[(pre_trial_measures['STORE_NBR'] == trial_s
control_sales = pre_trial_measures[(pre_trial_measures['STORE_NBR'].isin(cor
scaled_control_sales = apply_scaling_factor(measure_overtime_sales, trial_sa
scaled_control_sales
```

Out[48]:

	STORE_NBR	YEARMONTH	TOTAL_SALES	NUM_OF_CUSTOMERS	TRANSACTIO
2743	237	201807	1387.2	125	
2744	237	201808	1321.9	132	
2745	237	201809	1250.8	120	
2746	237	201810	1287.1	118	
2747	237	201811	1316.0	125	
2748	237	201812	1234.4	121	
2749	237	201901	1117.7	111	
2750	237	201902	1313.0	119	
2751	237	201903	1177.6	116	
2752	237	201904	1153.6	116	
2753	237	201905	1127.9	122	
2754	237	201906	1143.4	118	

In [49]:
```
# Calculate percentage difference between sales for scaled control & trial s
percentage_difference_sales = calculate_percentage_difference(scaled_control
percentage_difference_sales
```

Out[49]:

	YEARMONTH	CONTROL_SALES_SUM	TRIAL_SALES	PERCENTAGE_DIFFERENCE
0	201807	1374.4	1218.20	0.113650
1	201808	1309.7	1242.20	0.051539
2	201809	1239.3	1361.80	0.098846
3	201810	1275.2	1270.80	0.003450
4	201811	1303.9	1311.40	0.005752
5	201812	1223.0	1213.00	0.008177
6	201901	1107.4	1215.40	0.097526
7	201902	1300.9	1339.60	0.029749
8	201903	1166.7	1467.00	0.257393
9	201904	1143.0	1317.00	0.152231
10	201905	1117.5	1236.85	0.106801
11	201906	1132.8	1252.60	0.105756

Hypothesis Testing

In [50]:
```
perform_hypothesis_test(percentage_difference_sales)
```

```
H0: trial - pre_trial = 0 | Ha: trial - pre_trial =/= 0
Degree of Freedom: 7 | alpha = 0.05 | std_dev = 0.049079160365275945
T-Critical Value: 1.894578605061305
201902: Fail to Reject H0.
201903: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
201904: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
  YEARMONTH  CONTROL_SALES_SUM  TRIAL_SALES  PERCENTAGE_DIFFERENCE    tValue
7    201902             1300.9       1339.6               0.029749  0.606136
8    201903             1166.7       1467.0               0.257393  5.244439
9    201904             1143.0       1317.0               0.152231  3.101744
```

Trial Assessment

In [51]:
```
std_dev = percentage_difference_sales[percentage_difference_sales['YEARMONTH
trial_assessment = create_trial_assessment(measure_overtime_sales, std_dev,
trial_assessment
```

```
/var/folders/7m/wfwdc8s14pgch1s3hgyyg6080000gn/T/ipykernel_95955/778200519.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  past_data['TRANSACTION_MONTH'] = pd.to_datetime(past_data['YEARMONTH'].astype(str), format='%Y%m')
```

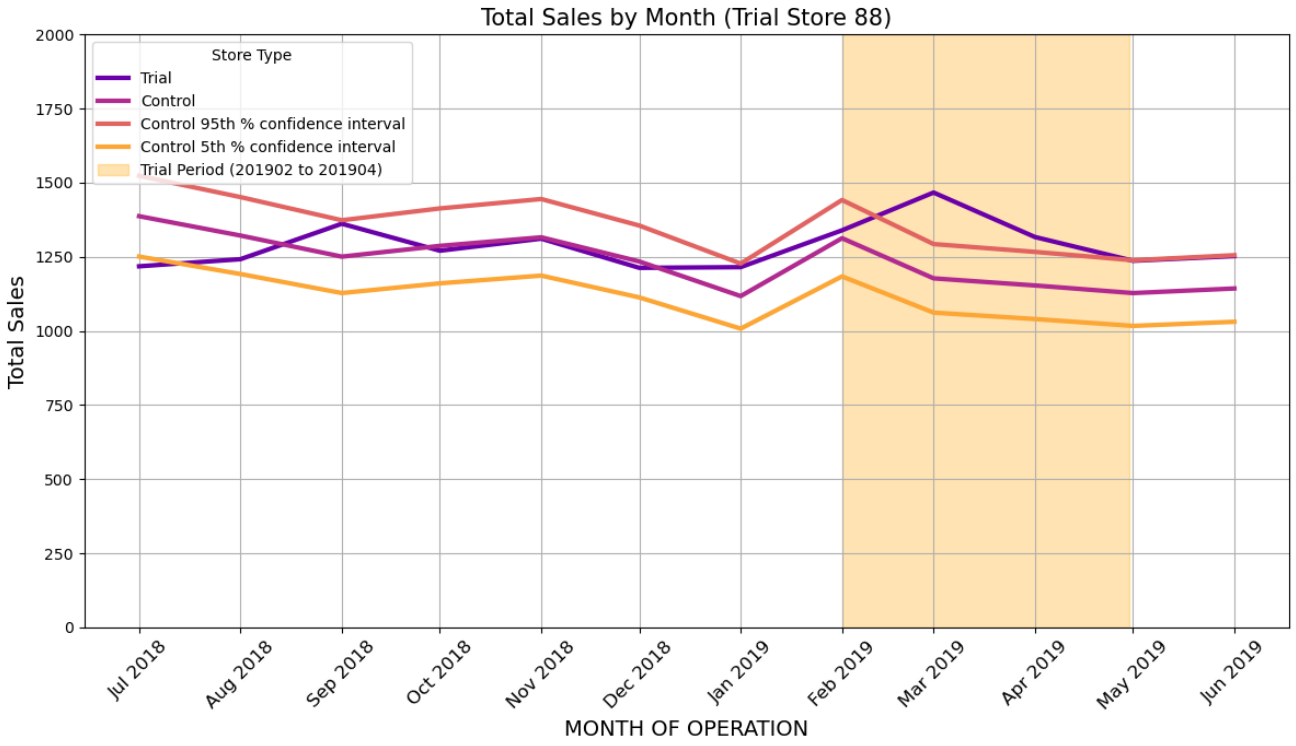
Out[51]:

TRANSACTION_MONTH	STORE_TYPE	YEARMONTH	TOTAL_SALES
2018-07-01	Trial	201807	1218.200000
2018-08-01	Trial	201808	1242.200000
2018-09-01	Trial	201809	1361.800000
2018-10-01	Trial	201810	1270.800000
2018-11-01	Trial	201811	1311.400000
2018-12-01	Trial	201812	1213.000000
2019-01-01	Trial	201901	1215.400000
2019-02-01	Trial	201902	1339.600000
2019-03-01	Trial	201903	1467.000000
2019-04-01	Trial	201904	1317.000000
2019-05-01	Trial	201905	1236.850000
2019-06-01	Trial	201906	1252.600000
2018-07-01	Control	201807	1387.200000
2018-08-01	Control	201808	1321.900000
2018-09-01	Control	201809	1250.800000
2018-10-01	Control	201810	1287.100000
2018-11-01	Control	201811	1316.000000
2018-12-01	Control	201812	1234.400000
2019-01-01	Control	201901	1117.700000
2019-02-01	Control	201902	1313.000000
2019-03-01	Control	201903	1177.600000
2019-04-01	Control	201904	1153.600000
2019-05-01	Control	201905	1127.900000
2019-06-01	Control	201906	1143.400000

2018-07-01	Control 95th % confidence interval	201807	1523.365223
2018-08-01	Control 95th % confidence interval	201808	1451.655484
2018-09-01	Control 95th % confidence interval	201809	1373.576428
2018-10-01	Control 95th % confidence interval	201810	1413.439575
2018-11-01	Control 95th % confidence interval	201811	1445.176350
2018-12-01	Control 95th % confidence interval	201812	1355.566631
2019-01-01	Control 95th % confidence interval	201901	1227.411555
2019-02-01	Control 95th % confidence interval	201902	1441.881875
2019-03-01	Control 95th % confidence interval	201903	1293.191238
2019-04-01	Control 95th % confidence interval	201904	1266.835439
2019-05-01	Control 95th % confidence interval	201905	1238.612770
2019-06-01	Control 95th % confidence interval	201906	1255.634224
2018-07-01	Control 5th % confidence interval	201807	1251.034777
2018-08-01	Control 5th % confidence interval	201808	1192.144516
2018-09-01	Control 5th % confidence interval	201809	1128.023572
2018-10-01	Control 5th % confidence interval	201810	1160.760425
2018-11-01	Control 5th % confidence interval	201811	1186.823650
2018-12-01	Control 5th % confidence interval	201812	1113.233369
2019-01-01	Control 5th % confidence interval	201901	1007.988445
2019-02-01	Control 5th % confidence interval	201902	1184.118125

	2019-03-01	Control 5th % confidence interval	201903	1062.008762
	2019-04-01	Control 5th % confidence interval	201904	1040.364561
	2019-05-01	Control 5th % confidence interval	201905	1017.187230
	2019-06-01	Control 5th % confidence interval	201906	1031.165776

In [52]:
```
# Plot the sales data for the trial store and control stores
plot_data(
    trial_assessment,
    metric='TOTAL_SALES',
    y_label='Total Sales',
    title=f'Total Sales by Month (Trial Store {trial_store})',
    ylim=(0,2000)
    )
```



The results show that the trial in store 88 is significantly different to its control store in the trial period as the trial store performance lies outside of the 5% to 95% confidence interval of the control store in two of the three trial months.

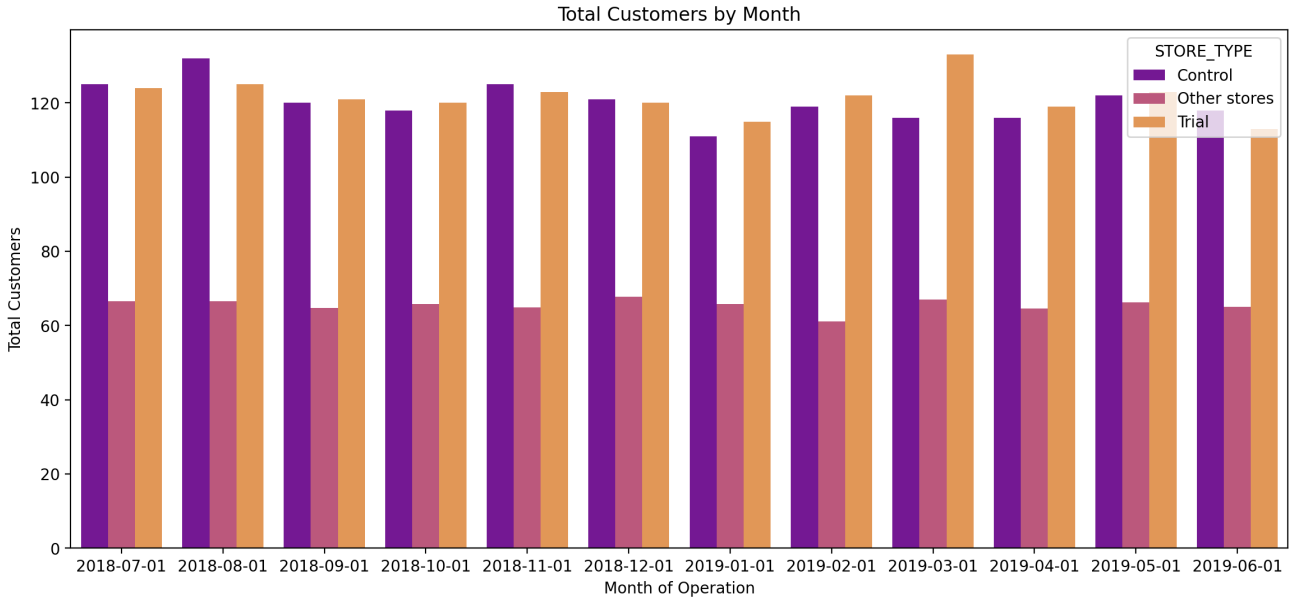
Customers Data Analysis

In [53]:
```
control_store = control_store_row.index.get_level_values('STORE_NBR_CONTROL'
# Assign store types ('Trial', 'Control', 'Other') to the measure_overtime_s
measure_overtime_customers = assign_store_types(measure_overtime.copy(), tri
```

```
# Group and calculate the mean of 'TOTAL_SALES' for each store type and mont
past_customers = measure_overtime_sales.groupby(['YEARMONTH', 'STORE_TYPE'])

# Convert 'YEARMONTH' to a datetime object
past_customers['TRANSACTION_MONTH'] = pd.to_datetime(past_customers['YEARMON

# Plot total sales by month
plot_metric_by_month(past_customers, 'NUM_OF_CUSTOMERS')
```



In [54]:
```
# Apply scaling factors for customers
trial_customers = pre_trial_measures[(pre_trial_measures['STORE_NBR'] == tri
control_customers = pre_trial_measures[(pre_trial_measures['STORE_NBR'].isir
scaled_control_customers = apply_scaling_factor(measure_overtime, trial_cust
scaled_control_customers
```

Out[54]:

	STORE_NBR	YEARMONTH	TOTAL_SALES	NUM_OF_CUSTOMERS	TRANSACTIO
2743	237	201807	1387.2	125	
2744	237	201808	1321.9	132	
2745	237	201809	1250.8	120	
2746	237	201810	1287.1	118	
2747	237	201811	1316.0	125	
2748	237	201812	1234.4	121	
2749	237	201901	1117.7	111	
2750	237	201902	1313.0	119	
2751	237	201903	1177.6	116	
2752	237	201904	1153.6	116	
2753	237	201905	1127.9	122	
2754	237	201906	1143.4	118	

In [55]:
```
# Calculate percentage difference
percentage_difference_customers = calculate_percentage_difference(scaled_cor
percentage_difference_customers
```

Out[55]:

	YEARMONTH	CONTROL_CUSTOMERS_SUM	TRIAL_CUSTOMERS	PERCENTAGE_DIF
0	201807	124.4	124	
1	201808	131.4	125	
2	201809	119.4	121	
3	201810	117.4	120	
4	201811	124.4	123	
5	201812	120.4	120	
6	201901	110.5	115	
7	201902	118.4	122	
8	201903	115.5	133	
9	201904	115.5	119	
10	201905	121.4	123	
11	201906	117.4	113	

Hypothesis Testing

In [56]:
```
perform_hypothesis_test(percentage_difference_customers)
```

```
H0: trial - pre_trial = 0 | Ha: trial - pre_trial =/= 0
Degree of Freedom: 7 | alpha = 0.05 | std_dev = 0.017967379944439074
T-Critical Value: 1.894578605061305
201902: Fail to Reject H0.
201903: Reject H0. There is sufficient evidence to conclude that sales/custo
mers differ.
201904: Fail to Reject H0.
  YEARMONTH  CONTROL_CUSTOMERS_SUM  TRIAL_CUSTOMERS  PERCENTAGE_DIFFERENCE
\
7    201902                  118.4              122               0.030405
8    201903                  115.5              133               0.151515
9    201904                  115.5              119               0.030303

     tValue
7  1.692256
8  8.432791
9  1.686558
```

Trial Assessment

In [57]:
```
# Trial Assessment:
std_dev = percentage_difference_customers[percentage_difference_customers['Y
trial_assessment = create_trial_assessment(measure_overtime_customers, std_d
trial_assessment
```

```
/var/folders/7m/wfwdc8s14pgch1s3hgyyg6080000gn/T/ipykernel_95955/778200519.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  past_data['TRANSACTION_MONTH'] = pd.to_datetime(past_data['YEARMONTH'].ast
ype(str), format='%Y%m')
```

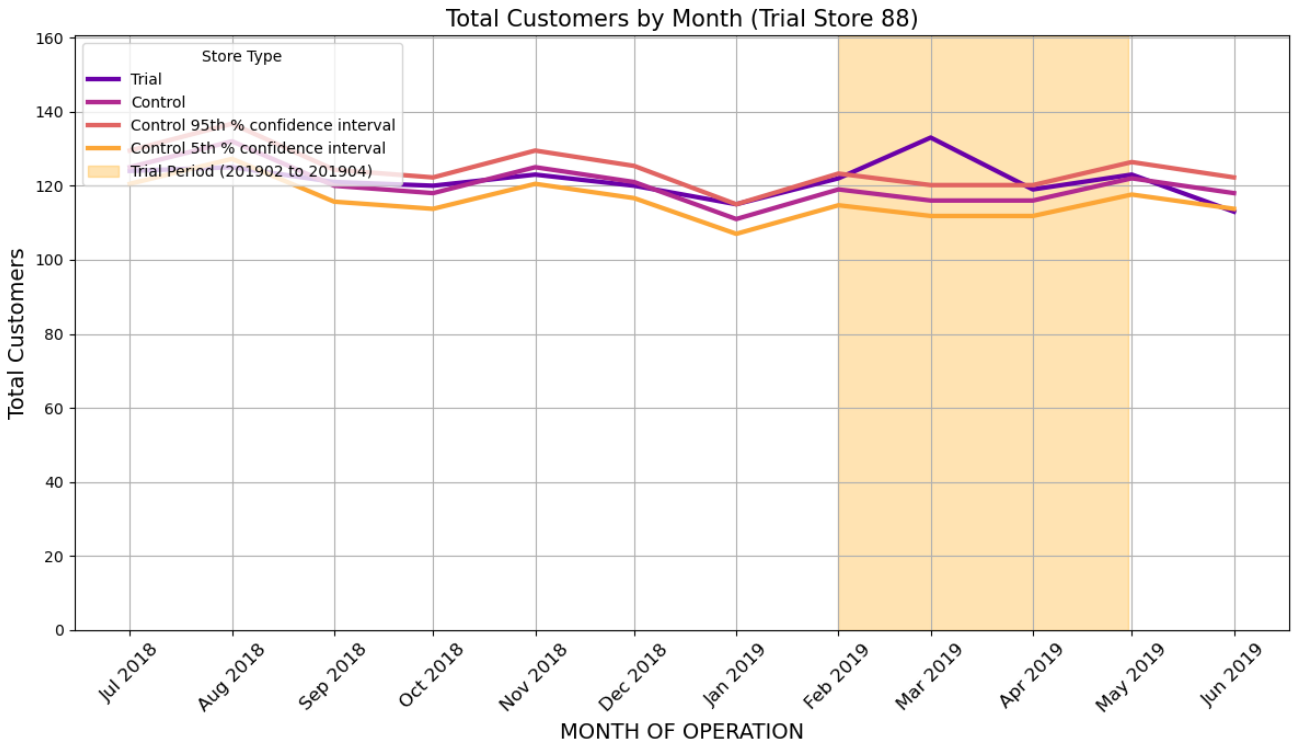
Out[57]:

TRANSACTION_MONTH	STORE_TYPE	YEARMONTH	NUM_OF_CUSTOMERS
2018-07-01	Trial	201807	124.000000
2018-08-01	Trial	201808	125.000000
2018-09-01	Trial	201809	121.000000
2018-10-01	Trial	201810	120.000000
2018-11-01	Trial	201811	123.000000
2018-12-01	Trial	201812	120.000000
2019-01-01	Trial	201901	115.000000

2019-02-01	Trial	201902	122.000000
2019-03-01	Trial	201903	133.000000
2019-04-01	Trial	201904	119.000000
2019-05-01	Trial	201905	123.000000
2019-06-01	Trial	201906	113.000000
2018-07-01	Control	201807	125.000000
2018-08-01	Control	201808	132.000000
2018-09-01	Control	201809	120.000000
2018-10-01	Control	201810	118.000000
2018-11-01	Control	201811	125.000000
2018-12-01	Control	201812	121.000000
2019-01-01	Control	201901	111.000000
2019-02-01	Control	201902	119.000000
2019-03-01	Control	201903	116.000000
2019-04-01	Control	201904	116.000000
2019-05-01	Control	201905	122.000000
2019-06-01	Control	201906	118.000000
2018-07-01	Control 95th % confidence interval	201807	129.491845
2018-08-01	Control 95th % confidence interval	201808	136.743388
2018-09-01	Control 95th % confidence interval	201809	124.312171
2018-10-01	Control 95th % confidence interval	201810	122.240302
2018-11-01	Control 95th % confidence interval	201811	129.491845
2018-12-01	Control 95th % confidence interval	201812	125.348106
2019-01-01	Control 95th % confidence interval	201901	114.988758
2019-02-01	Control 95th % confidence interval	201902	123.276236
2019-03-01	Control 95th % confidence interval	201903	120.168432

2019-04-01	Control 95th % confidence interval	201904	120.168432
2019-05-01	Control 95th % confidence interval	201905	126.384041
2019-06-01	Control 95th % confidence interval	201906	122.240302
2018-07-01	Control 5th % confidence interval	201807	120.508155
2018-08-01	Control 5th % confidence interval	201808	127.256612
2018-09-01	Control 5th % confidence interval	201809	115.687829
2018-10-01	Control 5th % confidence interval	201810	113.759698
2018-11-01	Control 5th % confidence interval	201811	120.508155
2018-12-01	Control 5th % confidence interval	201812	116.651894
2019-01-01	Control 5th % confidence interval	201901	107.011242
2019-02-01	Control 5th % confidence interval	201902	114.723764
2019-03-01	Control 5th % confidence interval	201903	111.831568
2019-04-01	Control 5th % confidence interval	201904	111.831568
2019-05-01	Control 5th % confidence interval	201905	117.615959
2019-06-01	Control 5th % confidence interval	201906	113.759698

In [58]:
```
# Plot the data for NUM_OF_CUSTOMERS
plot_data(
        trial_assessment,
        metric='NUM_OF_CUSTOMERS',
        y_label='Total Customers',
        title=f'Total Customers by Month (Trial Store {trial_store})'
    )
```



Overall Sales Performance

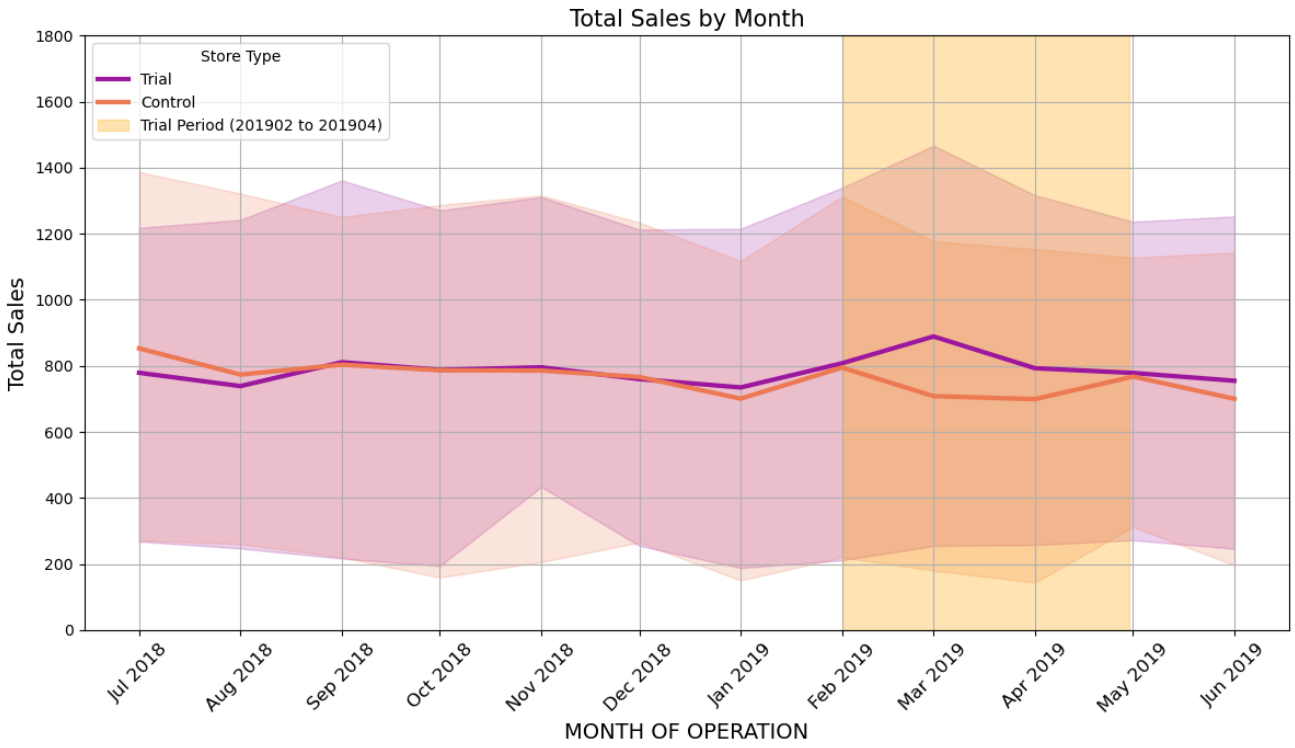
In [59]:
```
trial_control = [77, 233, 86, 155, 88, 237]

# Filter the data for the selected stores
all_sales = measure_overtime_sales[measure_overtime_sales['STORE_NBR'].isin(

all_sales['TRANSACTION_MONTH'] = pd.to_datetime(all_sales['YEARMONTH'].astyp

all_sales['STORE_TYPE'] = all_sales['STORE_NBR'].apply(
    lambda x: 'Trial' if x in [77, 86, 88] else 'Control')

plot_data(
    all_sales,
    metric='TOTAL_SALES',
    y_label='Total Sales',
    title=f'Total Sales by Month', ylim=(0,1800)
)
```



Summary of Insights

1. In Pre-Trial period, the average sales per store per month is $586, along with an average of 1.14 transactions per customer and 3.81 per chip unit. During the trial period (from February to April, 2019), the average sales per store per month is $573, indicating substantial impact on the stores' transaction values. Even though with just 3 months of trial, multiple Key Performance Indicators, such as number of customers, transaction per customer, chips per customer and average price per unit remains relatively close to the 7 months of data pre-trial.
2. Upon the data wrangling and analysis process, we have found the following trial - control store pairs matched based on performanced metrics (Total Sales & Number of Customers):

- Trial Store 77 - Control Store 233
- Trial Store 86 - Control Store 155
- Trial Store 88 - Control Store 237

3. For Trial Store 77: Total Sales and Number of Customers are significantly increased during the trial period in at least two of the three trial months (March & April).
4. For Trial Store 86: Even though the number of customers is significantly higher in all of the three months, the Total Sales were not significantly higher in all 3 months (only February and March). We should check with the Category Manager if there were special deals in the trial store that were may have resulted in lower prices, impacting the results.