QUANTIUM VIRTUAL INTERNSHIP TASK 2

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df=pd.read_csv('C:/Users/ACER/Downloads/QVI_data.csv')
df.head(20)
```

Out[3]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PR
0	1000	10/17/2018	1	1	5	Natural Chip Compny SeaSalt175g	
1	1002	9/16/2018	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	
2	1003	3/7/2019	1	3	52	Grain Waves Sour Cream&Chives 210G	
3	1003	3/8/2019	1	4	106	Natural ChipCo Hony Soy Chckn175g	
4	1004	11/2/2018	1	5	96	WW Original Stacked Chips 160g	
5	1005	12/28/2018	1	6	86	Cheetos Puffs 165g	
6	1007	12/4/2018	1	7	49	Infuzions SourCream&Herbs Veg Strws 110g	
7	1007	12/5/2018	1	8	10	RRD SR Slow Rst Pork Belly 150g	
8	1009	11/20/2018	1	9	20	Doritos Cheese Supreme 330g	
9	1010	9/9/2018	1	10	51	Doritos Mexicana 170g	
10	1010	12/14/2018	1	11	59	Old El Paso Salsa Dip Tomato Med 300g	
11	1011	7/29/2018	1	12	84	GrnWves Plus Btroot & Chilli Jam 180g	
12	1011	11/8/2018	1	13	59	Old El Paso Salsa Dip Tomato Med 300g	
13	1011	12/1/2018	1	14	49	Infuzions SourCream&Herbs Veg Strws 110g	
14	1011	12/19/2018	1	15	1	Smiths Crinkle Cut Chips Barbecue 170g	

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PR
15	1012	3/15/2019	1	16	20	Doritos Cheese Supreme 330g	
16	1012	6/19/2019	1	17	3	Kettle Sensations Camembert & Fig 150g	
17	1013	3/4/2019	1	18	93	Doritos Corn Chip Southern Chicken 150g	
18	1013	3/7/2019	1	19	91	CCs Tasty Cheese 175g	
19	1016	4/19/2019	1	20	74	Tostitos Splash Of Lime 175g	

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264834 entries, 0 to 264833
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   LYLTY_CARD_NBR    264834 non-null  int64  
 1   DATE              264834 non-null  object 
 2   STORE_NBR         264834 non-null  int64  
 3   TXN_ID            264834 non-null  int64  
 4   PROD_NBR          264834 non-null  int64  
 5   PROD_NAME         264834 non-null  object 
 6   PROD_QTY          264834 non-null  int64  
 7   TOT_SALES         264834 non-null  float64
 8   PACK_SIZE         264834 non-null  int64  
 9   BRAND             264834 non-null  object 
 10  LIFESTAGE         264834 non-null  object 
 11  PREMIUM_CUSTOMER  264834 non-null  object 
dtypes: float64(1), int64(6), object(5)
memory usage: 24.2+ MB
```

In [5]:
```
df.describe()
```

Out[5]:

	LYLTY_CARD_NBR	STORE_NBR	TXN_ID	PROD_NBR	PROD_QTY	T
count	2.648340e+05	264834.000000	2.648340e+05	264834.000000	264834.000000	2648
mean	1.355488e+05	135.079423	1.351576e+05	56.583554	1.905813	
std	8.057990e+04	76.784063	7.813292e+04	32.826444	0.343436	
min	1.000000e+03	1.000000	1.000000e+00	1.000000	1.000000	
25%	7.002100e+04	70.000000	6.760050e+04	28.000000	2.000000	
50%	1.303570e+05	130.000000	1.351365e+05	56.000000	2.000000	
75%	2.030940e+05	203.000000	2.026998e+05	85.000000	2.000000	
max	2.373711e+06	272.000000	2.415841e+06	114.000000	5.000000	

In [6]:
```
df['DATE']=pd.to_datetime(df['DATE'])
df['YEARMONTH']=df['DATE'].dt.strftime('%Y%m')
df.head()
```

Out[6]:

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

In [7]:
```
metrics=df.groupby(['STORE_NBR','YEARMONTH']).agg(
    monthly_sales=('TOT_SALES','sum'),
    monthly_customers=('LYLTY_CARD_NBR',pd.Series.nunique),
    transactions=('TXN_ID','count')
).reset_index()
metrics['chips_per_customer']=metrics['transactions']/metrics['monthly_customers']
metrics['avg_price_per_unit']=metrics['monthly_sales']/metrics['transactions']
metrics.head(8)
```

Out[7]:

	STORE_NBR	YEARMONTH	monthly_sales	monthly_customers	transactions	chips_per_cu
0	1	201807	206.9	49	52	1
1	1	201808	176.1	42	43	1
2	1	201809	278.8	59	62	1
3	1	201810	188.1	44	45	1
4	1	201811	192.6	46	47	1
5	1	201812	189.6	42	47	1
6	1	201901	154.8	35	36	1
7	1	201902	225.4	52	55	1

In [8]:
```
pre_trial_months=['201807','201808','201809','201810','201811','201812','201901','2
pre_trial=metrics[metrics['YEARMONTH'].isin(pre_trial_months)]
store_month_count=pre_trial.groupby('STORE_NBR')['YEARMONTH'].nunique().reset_index
valid_stores=store_month_count[store_month_count['YEARMONTH']==len(pre_trial_months
pre_trial=pre_trial[pre_trial['STORE_NBR'].isin(valid_stores)]
pre_trial.head(10)
```

Out[8]:

	STORE_NBR	YEARMONTH	monthly_sales	monthly_customers	transactions	chips_per_
0	1	201807	206.9	49	52	
1	1	201808	176.1	42	43	
2	1	201809	278.8	59	62	
3	1	201810	188.1	44	45	
4	1	201811	192.6	46	47	
5	1	201812	189.6	42	47	
6	1	201901	154.8	35	36	
7	1	201902	225.4	52	55	
12	2	201807	150.8	39	41	
13	2	201808	193.8	39	43	

In [9]:
```
trial_stores=[77,86,88]
trial_data=pre_trial[pre_trial['STORE_NBR'].isin(trial_stores)]
trial_data
```

Out[9]:

	STORE_NBR	YEARMONTH	monthly_sales	monthly_customers	transactions	chips_pe
880	77	201807	296.80	51	55	
881	77	201808	255.50	47	48	
882	77	201809	225.20	42	44	
883	77	201810	204.50	37	38	
884	77	201811	245.30	41	44	
885	77	201812	267.30	46	49	
886	77	201901	204.40	35	39	
887	77	201902	235.00	45	45	
977	86	201807	892.20	99	126	
978	86	201808	764.05	94	112	
979	86	201809	914.60	103	129	
980	86	201810	948.40	109	138	
981	86	201811	918.00	100	127	
982	86	201812	841.20	98	120	
983	86	201901	841.40	94	130	
984	86	201902	913.20	107	139	
1001	88	201807	1310.00	129	153	
1002	88	201808	1323.80	131	160	
1003	88	201809	1423.00	124	159	
1004	88	201810	1352.40	123	158	
1005	88	201811	1382.80	130	157	
1006	88	201812	1325.20	126	149	
1007	88	201901	1266.40	117	146	
1008	88	201902	1370.20	124	154	

In [10]:

```
def calc_correlation(df,metric_col,trial_stores):
    """Calculates the correlation between all trial stores and all control stores u
    wide_df= df.pivot(index='YEARMONTH',columns='STORE_NBR',values=metric_col)
    corr_matrix=wide_df.corr()
    final_rankings=[]
    #Get a list of all unique store IDs
    all_stores=corr_matrix.index.tolist()
    #identify control stores(all stores not in trial list)
    control_stores=[s for s in all_stores if s not in trial_stores]
    #Loop through each of the trial stores
```

```
    for trial_store in trial_stores:
        corr_to_trial=corr_matrix[trial_store].copy()
    #filter the series to include only the control stores
    results=corr_to_trial[control_stores]
    #convert it into DataFrame
    ranking_df=results.sort_values(ascending=False).to_frame(name='Correlation')
    ranking_df.index_name='Control_Store'
    ranking_df['Trial_Store']=trial_store
    ranking_df=ranking_df.reset_index()
    final_rankings.append(ranking_df)
    return pd.concat(final_rankings,ignore_index=True)
```

In [11]:

```
trial_store_list=[77]
sales_ranking=calc_correlation(pre_trial,'monthly_sales',trial_store_list)
print(sales_ranking)
customer_ranking=calc_correlation(pre_trial,'monthly_customers',trial_store_list)
print(customer_ranking)
transaction_ranking=calc_correlation(pre_trial,'transactions',trial_store_list)
print(transaction_ranking)
```

```
     STORE_NBR  Correlation  Trial_Store
0          233     0.894375           77
1           17     0.843806           77
2          119     0.831832           77
3           71     0.792931           77
4          157     0.722332           77
..         ...          ...          ...
254        169    -0.633658           77
255        242    -0.642253           77
256         24    -0.645190           77
257         19    -0.657963           77
258        244    -0.674915           77

[259 rows x 3 columns]
     STORE_NBR  Correlation  Trial_Store
0          233     0.990542           77
1          119     0.977709           77
2           41     0.811844           77
3            3     0.756913           77
4           17     0.741196           77
..         ...          ...          ...
254        186    -0.644356           77
255        267    -0.651727           77
256        102    -0.662607           77
257         54    -0.680967           77
258          9    -0.700131           77

[259 rows x 3 columns]
     STORE_NBR  Correlation  Trial_Store
0          233     0.953596           77
1          119     0.896548           77
2           17     0.829773           77
3           41     0.810380           77
4          157     0.808520           77
..         ...          ...          ...
254        172    -0.568064           77
255        138    -0.568942           77
256        169    -0.665556           77
257        147    -0.700243           77
258        242    -0.759368           77

[259 rows x 3 columns]
```

In [12]:

```python
def calc_magnitude_distance(df,metric_col,trial_stores):
    """Calculate the standardized magnitude distance (average absolute difference
    trial stores and all control stores using pivot method"""
    wide_df=df.pivot(index='YEARMONTH',columns='STORE_NBR',values=metric_col)
    distance_data=[]
    control_stores=[col for col in wide_df.columns if col not in trial_stores]
    #calculate absolute distance for all pairs
    for trial_store in trial_stores:
        #Subtract the trial's store performance form all the control stores ,
        #take the absolute value and then calulate the mean
        distances=wide_df[control_stores].sub(wide_df[trial_store],axis=0).abs().m
        #store the results with the trial store identifier
        temp_df=pd.DataFrame({
```

```
            'Trial_Store':trial_store,
            'Control_Store':distances.index,
            'Avg_mag_distance':distances.values
        })
        distance_data.append(temp_df)
    #combine results into one dataframe
    avg_distance_df=pd.concat(distance_data).reset_index(drop=True)
    #Standardize the results (min-Max Scaling)
    min_dist= avg_distance_df['Avg_mag_distance'].min()
    max_dist=avg_distance_df['Avg_mag_distance'].max()
    #apply standardization: 1-(measure-min)/(max-min)
    #Closure to 1 is better(min distance).
    range_diff=max_dist-min_dist if max_dist !=min_dist else 1e-9
    avg_distance_df['Magnitude_Measure']=1-(
        (avg_distance_df['Avg_mag_distance']-
         min_dist)/ range_diff
    )
    return avg_distance_df[['Trial_Store','Control_Store','Magnitude_Measure']]
```

In [13]:
```
sales_mag_ranking= calc_magnitude_distance(pre_trial,'monthly_sales',trial_store_li
print(sales_mag_ranking.sort_values(by='Magnitude_Measure',ascending=False).head())
```

	Trial_Store	Control_Store	Magnitude_Measure
220	77	233	1.000000
241	77	255	0.991664
42	77	46	0.988914
179	77	188	0.988532
202	77	214	0.986642

In [14]:
```
customer_mag_ranking= calc_magnitude_distance(pre_trial,'monthly_customers',trial_s
print(customer_mag_ranking.sort_values(by='Magnitude_Measure',ascending=False).head
```

	Trial_Store	Control_Store	Magnitude_Measure
220	77	233	1.000000
38	77	41	0.976190
15	77	17	0.970588
107	77	115	0.967787
42	77	46	0.959384

In [15]:
```
score_sales=pd.merge(sales_ranking,sales_mag_ranking,on=['Trial_Store'],suffixes=('
#calculate combined sales score : 0.5*Correlation + 0.5*Magnitude_measure
corr_weight = 0.5
mag_weight = 1-corr_weight
score_sales['scoreSales']=(score_sales['Correlation']*corr_weight)+(score_sales['Ma
print(score_sales)
```

```
       STORE_NBR  Correlation  Trial_Store  Control_Store  Magnitude_Measure  \
0            233     0.894375           77              1           0.969408
1            233     0.894375           77              2           0.944220
2            233     0.894375           77              3           0.293957
3            233     0.894375           77              4           0.157343
4            233     0.894375           77              5           0.533637
...          ...          ...          ...            ...                ...
67076        244    -0.674915           77            268           0.970305
67077        244    -0.674915           77            269           0.411204
67078        244    -0.674915           77            270           0.416869
67079        244    -0.674915           77            271           0.528901
67080        244    -0.674915           77            272           0.886906

       scoreSales
0        0.931891
1        0.919298
2        0.594166
3        0.525859
4        0.714006
...           ...
67076    0.147695
67077   -0.131856
67078   -0.129023
67079   -0.073007
67080    0.105995

[67081 rows x 6 columns]
```

In [16]:
```
score_customer=pd.merge(customer_ranking,customer_mag_ranking,on=['Trial_Store'],su
#calculate combined sales score : 0.5*Correlation + 0.5*Magnitude_measure
corr_weight = 0.5
mag_weight = 1-corr_weight
score_customer['scoreNcust']=(score_customer['Correlation']*corr_weight)+(score_cus
print(score_customer)
```

```
       STORE_NBR  Correlation  Trial_Store  Control_Store  Magnitude_Measure  \
0            233     0.990542           77              1           0.941176
1            233     0.990542           77              2           0.913165
2            233     0.990542           77              3           0.284314
3            233     0.990542           77              4           0.172269
4            233     0.990542           77              5           0.455182
...          ...          ...          ...            ...                ...
67076          9    -0.700131           77            268           0.936975
67077          9    -0.700131           77            269           0.299720
67078          9    -0.700131           77            270           0.329132
67079          9    -0.700131           77            271           0.455182
67080          9    -0.700131           77            272           0.957983

       scoreNcust
0        0.965859
1        0.951854
2        0.637428
3        0.581406
4        0.722862
...           ...
67076    0.118422
67077   -0.200205
67078   -0.185499
67079   -0.122474
67080    0.128926

[67081 rows x 6 columns]
```

In [17]:
```
final_score_ranking=pd.merge(score_sales,score_customer,on=['Control_Store','Trial_
print(final_score_ranking)
```

```
          STORE_NBR_sales  Correlation_sales  Trial_Store  Control_Store  \
0                     233           0.894375           77              1
1                     233           0.894375           77              1
2                     233           0.894375           77              1
3                     233           0.894375           77              1
4                     233           0.894375           77              1
...                   ...                ...          ...            ...
17373974              244          -0.674915           77            272
17373975              244          -0.674915           77            272
17373976              244          -0.674915           77            272
17373977              244          -0.674915           77            272
17373978              244          -0.674915           77            272

          Magnitude_Measure_sales  scoreSales  STORE_NBR_cust  \
0                        0.969408    0.931891             233
1                        0.969408    0.931891             119
2                        0.969408    0.931891              41
3                        0.969408    0.931891               3
4                        0.969408    0.931891              17
...                           ...         ...             ...
17373974                 0.886906    0.105995             186
17373975                 0.886906    0.105995             267
17373976                 0.886906    0.105995             102
17373977                 0.886906    0.105995              54
17373978                 0.886906    0.105995               9

          Correlation_cust  Magnitude_Measure_cust  scoreNcust
0                 0.990542                0.941176    0.965859
1                 0.977709                0.941176    0.959443
2                 0.811844                0.941176    0.876510
3                 0.756913                0.941176    0.849045
4                 0.741196                0.941176    0.841186
...                    ...                     ...         ...
17373974         -0.644356                0.957983    0.156814
17373975         -0.651727                0.957983    0.153128
17373976         -0.662607                0.957983    0.147688
17373977         -0.680967                0.957983    0.138508
17373978         -0.700131                0.957983    0.128926

[17373979 rows x 10 columns]
```

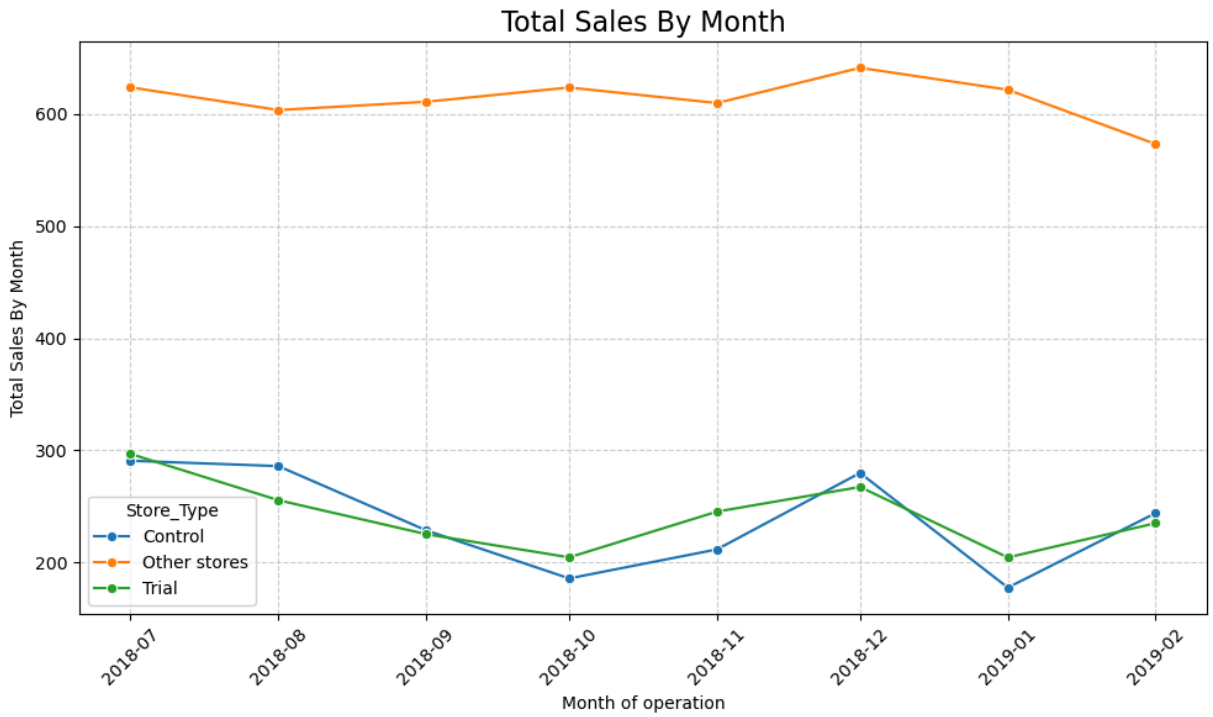
In [18]:
```
#calculate the final composite score using the suffixed column names
final_score_ranking['finalControlScore']=(final_score_ranking['scoreSales']*0.5)+(f
final_ranking=final_score_ranking.sort_values(by=['Trial_Store','finalControlScore'
print(final_ranking[['Trial_Store','Control_Store','finalControlScore']].head(10))
```

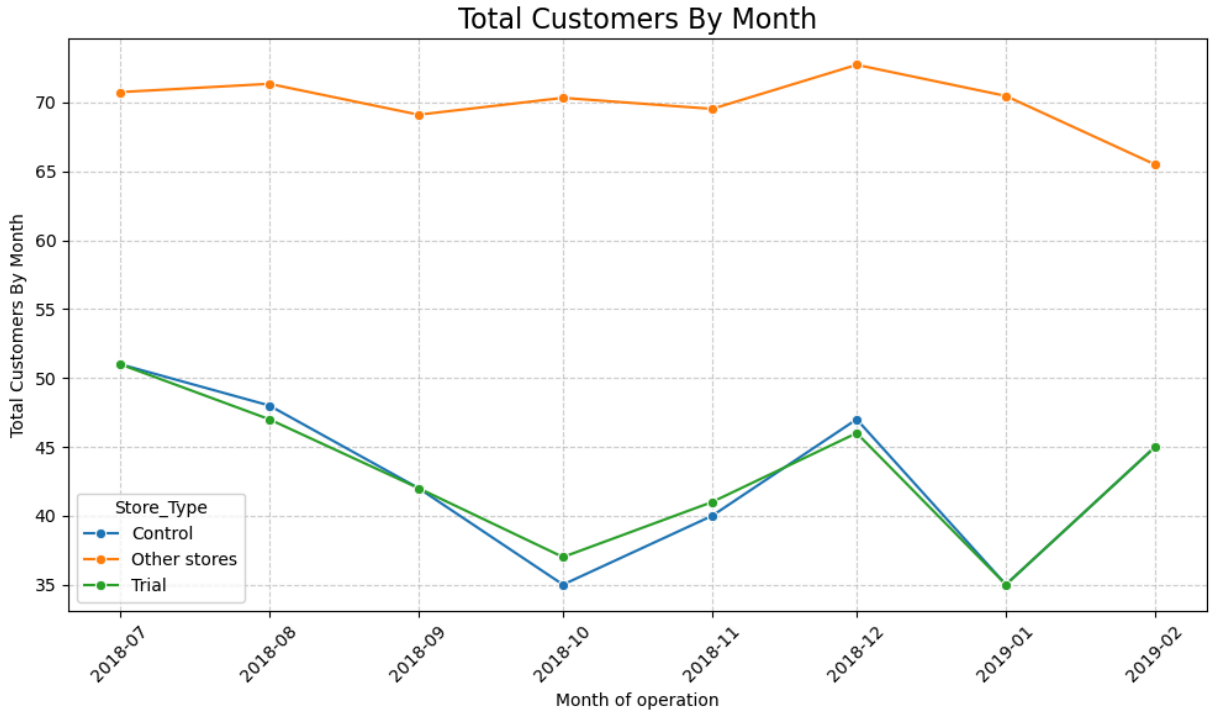
	Trial_Store	Control_Store	finalControlScore
0	77	233	0.971229
1	77	233	0.968021
2	77	41	0.960578
3	77	233	0.958587
4	77	46	0.958304
5	77	41	0.957370
6	77	53	0.955894
7	77	233	0.955593
8	77	188	0.955407
9	77	233	0.955379

In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
def prepare_plot(metrics,trial_store_nbr,control_store_nbr):
    df_plot=metrics.copy()
    df_plot['STORE_NBR']=df_plot['STORE_NBR'].astype(int)
    conditions=[(df_plot['STORE_NBR']==trial_store_nbr),(df_plot['STORE_NBR']== con
    choices=['Trial','Control']
    df_plot['Store_Type']= np.select(conditions,choices,default='Other stores')
    df_plot=df_plot[df_plot['YEARMONTH'].astype(int)<201903]
    plot_data=df_plot.groupby(['YEARMONTH','Store_Type']).agg(
        total_sales=('monthly_sales','mean'),
        total_customers=('monthly_customers','mean')
    ).reset_index()
    plot_data['Transaction_Month']=pd.to_datetime(plot_data['YEARMONTH'].astype(str
    return plot_data
trial_store=77
control_store=233
plot_df= prepare_plot(metrics,trial_store,control_store)
print(plot_df)
```

	YEARMONTH	Store_Type	total_sales	total_customers	Transaction_Month
0	201807	Control	290.700000	51.000000	2018-07-01
1	201807	Other stores	623.817424	70.750000	2018-07-01
2	201807	Trial	296.800000	51.000000	2018-07-01
3	201808	Control	285.900000	48.000000	2018-08-01
4	201808	Other stores	603.600192	71.352490	2018-08-01
5	201808	Trial	255.500000	47.000000	2018-08-01
6	201809	Control	228.600000	42.000000	2018-09-01
7	201809	Other stores	610.947328	69.110687	2018-09-01
8	201809	Trial	225.200000	42.000000	2018-09-01
9	201810	Control	185.700000	35.000000	2018-10-01
10	201810	Other stores	623.671103	70.334601	2018-10-01
11	201810	Trial	204.500000	37.000000	2018-10-01
12	201811	Control	211.600000	40.000000	2018-11-01
13	201811	Other stores	609.835115	69.534351	2018-11-01
14	201811	Trial	245.300000	41.000000	2018-11-01
15	201812	Control	279.800000	47.000000	2018-12-01
16	201812	Other stores	641.250192	72.731801	2018-12-01
17	201812	Trial	267.300000	46.000000	2018-12-01
18	201901	Control	177.500000	35.000000	2019-01-01
19	201901	Other stores	621.687356	70.471264	2019-01-01
20	201901	Trial	204.400000	35.000000	2019-01-01
21	201902	Control	244.000000	45.000000	2019-02-01
22	201902	Other stores	573.229008	65.492366	2019-02-01
23	201902	Trial	235.000000	45.000000	2019-02-01

In [20]:
```
def visualize_trend(plot_data, metric_col,title):
    """Creates a line plot to visuallise the trend of a metric over time"""
    y_column = metric_col
    y_label = title
    plt.figure(figsize=(10,6))
    sns.lineplot(
        data=plot_data,
        x='Transaction_Month',
        y=y_column,
        hue='Store_Type',
        errorbar= None,
        marker='o'
    )
    plt.title(title,fontsize=16)
    plt.xlabel("Month of operation")
    plt.ylabel(y_label)
    plt.grid(True,linestyle='--',alpha=0.6)
    plt.legend(title='Store_Type')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
visualize_trend(plot_df,'total_sales',"Total Sales By Month")
visualize_trend(plot_df,'total_customers',"Total Customers By Month")
```





In [21]:
```
def calculate_scaling_factor(pre_trial,trial_store,control_store):
    """Calculates the scaling factor by comparing pre-trial sums."""
    #filter for the specific months
    df_scale=pre_trial[pre_trial['YEARMONTH'].astype(int)<=201902]
    #get trial stores sum for the scaling month
    trial_sales=df_scale[df_scale['STORE_NBR']==trial_store]['monthly_sales'].sum()
    #get control store sales sum for scaling month
    control_sales=df_scale[df_scale['STORE_NBR']==control_store]['monthly_sales'].s
    #Scaling_factor = Trial_sales/Control_sales
    if control_sales !=0:
        scaling_factor=trial_sales/control_sales
    else :
```

```
        scaling_factor=1.0
    return scaling_factor
scaling_factor=calculate_scaling_factor(pre_trial,77,233)
print(scaling_factor)
```

```
1.0158630108204645
```

In [22]:
```
def apply_scaling_factor(metrics,control_stores,scaling_factor):
    """Applies the scaling factore to the control store's 'monthly_sales' column.""
    #create a copy to avoid warning
    df_scaled=metrics.copy()
    #apply scaling only to the control's data
    condition = (df_scaled['STORE_NBR']==control_store)
    #calc the scaled sales clm
    df_scaled.loc[condition,'scaledControlSales']= df_scaled.loc[condition,'monthly
    #for the trial store and other stores , set scaledControlStores to the original
    #we only care about the control store for this column
    df_scaled.loc[~condition,'scaledControlSales']=df_scaled.loc[~condition,'monthl
    return df_scaled
metrics_scaled_df=apply_scaling_factor(metrics,233,scaling_factor)
print(metrics_scaled_df)
```

```
      STORE_NBR YEARMONTH  monthly_sales  monthly_customers  transactions  \
0             1    201807          206.9                 49            52
1             1    201808          176.1                 42            43
2             1    201809          278.8                 59            62
3             1    201810          188.1                 44            45
4             1    201811          192.6                 46            47
...         ...       ...            ...                ...           ...
3164        272    201902          395.5                 45            48
3165        272    201903          442.3                 50            53
3166        272    201904          445.1                 54            56
3167        272    201905          314.6                 34            40
3168        272    201906          312.1                 34            37

      chips_per_customer  avg_price_per_unit  scaledControlSales
0               1.061224            3.978846               206.9
1               1.023810            4.095349               176.1
2               1.050847            4.496774               278.8
3               1.022727            4.180000               188.1
4               1.021739            4.097872               192.6
...                  ...                 ...                 ...
3164            1.066667            8.239583               395.5
3165            1.060000            8.345283               442.3
3166            1.037037            7.948214               445.1
3167            1.176471            7.865000               314.6
3168            1.088235            8.435135               312.1

[3169 rows x 8 columns]
```

In [23]:
```
def calc_percentage_diff(metrics_scaled_df,trial_store,control_store):
    """Calculate the percentage difference using filtering and merging"""
    trial_months = ['201902','201903','201904']
    control_data=metrics_scaled_df[(metrics_scaled_df['STORE_NBR']==control_store)
    (metrics_scaled_df['YEARMONTH'].isin(trial_months))].copy()
    control_data= control_data[['YEARMONTH','scaledControlSales']].rename(
    columns={
```

```
    'scaledControlSales':'ScaledControlSales'}
    )
    trial_data = metrics_scaled_df[(metrics_scaled_df['STORE_NBR']==trial_store) &
    (metrics_scaled_df['YEARMONTH'].isin(trial_months))].copy()
    trial_data= trial_data[['YEARMONTH','monthly_sales']].rename(
    columns={
    'monthly_sales':'TrialSales_Actual'}
    )
    final_comparison= pd.merge(control_data,trial_data,on='YEARMONTH',how='inner')
    final_comparison['percentage_diff']= (final_comparison['TrialSales_Actual']-fin
    return final_comparison
final_diff_table=calc_percentage_diff(metrics_scaled_df,77,233)
print(final_diff_table)
```

```
   YEARMONTH  ScaledControlSales  TrialSales_Actual  percentage_diff
0     201902          247.870575              235.0        -0.051925
1     201903          202.258325              278.5         0.376952
2     201904          161.115874              263.5         0.635469
```

In [24]:
```
final_diff_table['YEARMONTH']=final_diff_table['YEARMONTH'].astype(int)
stdDev=final_diff_table[final_diff_table['YEARMONTH']>=201902]['percentage_diff'].s
print(f"Standard Deviation of pre-trial percentage diff:{stdDev:.4f}")
```

```
Standard Deviation of pre-trial percentage diff:0.3472
```

In [170…
```
from scipy.stats import t
degreesOfFreedom=7 #8-1(trial period months)
trial_data1= final_diff_table.copy()
#Calculate the t-value: t=(x-mu)/stdDev since mu(null hypothesis is 0:
#trial_data['tvalue']=trial_data['percentage_diff']/stdDev
trial_data1['tValue'] = np.where(stdDev!=0,trial_data1['percentage_diff']/stdDev,0)
#find the 95th percentile of the t-distribution
t_percentile_95=t.ppf(0.95,degreesOfFreedom)
print(f"(95th percentile t-value (df=7): {t_percentile_95:.4f}")
print("\nTrial Month t-values:")
print(trial_data1[['YEARMONTH','percentage_diff','tValue']])
```

```
(95th percentile t-value (df=7): 1.8946

Trial Month t-values:
   YEARMONTH  percentage_diff    tValue
0     201902        -0.051925 -0.149554
1     201903         0.376952  1.085700
2     201904         0.635469  1.830282
```

In [26]:
```
def create_assessment_data(metrics_scaled_df,stdDev):
    """Calculates the 5th and 95th percentile sales band for the control stores."""
    control_store=233
    control_base=metrics_scaled_df[metrics_scaled_df['STORE_NBR']== control_store].
    #control 95th percentile (upper bound)
    pastSalesControls95= control_base.copy()
    #formula:totSales*(1+stdDev*2)
    pastSalesControls95['monthly_sales']=pastSalesControls95['monthly_sales']*(1+st
    pastSalesControls95['Store_Type']="Control 95th % confidence Interval"
    #control 5th percentile (Lower Bound)
    pastSalesControls5= control_base.copy()
    pastSalesControls5['monthly_sales']=pastSalesControls95['monthly_sales']*(1-std
```

```
    pastSalesControls5['Store_Type']="Control 5th % confidence Interval"
    trial_assessment= pd.concat([pastSalesControls95,pastSalesControls5],ignore_ind
    trial_assessment['Transaction_Month']=pd.to_datetime(
        trial_assessment['YEARMONTH'].astype(str),format='%Y%m')
    return trial_assessment
trial_ass_df= create_assessment_data(metrics_scaled_df,stdDev)
print(trial_ass_df)
```

	STORE_NBR	YEARMONTH	monthly_sales	monthly_customers	transactions	\
0	233	201807	492.560523	51	54	
1	233	201808	484.427429	48	50	
2	233	201809	387.338616	42	45	
3	233	201810	314.649085	35	36	
4	233	201811	358.533907	40	41	
5	233	201812	474.091621	47	50	
6	233	201901	300.755049	35	35	
7	233	201902	413.432293	45	47	
8	233	201903	337.353974	40	41	
9	233	201904	268.730991	30	33	
10	233	201905	583.549515	57	62	
11	233	201906	374.461216	41	41	
12	233	201807	150.529134	51	54	
13	233	201808	148.043617	48	50	
14	233	201809	118.372756	42	45	
15	233	201810	96.158446	35	36	
16	233	201811	109.569882	40	41	
17	233	201812	144.884939	47	50	
18	233	201901	91.912354	35	35	
19	233	201902	126.347123	45	47	
20	233	201903	103.097181	40	41	
21	233	201904	82.125630	30	33	
22	233	201905	178.335857	57	62	
23	233	201906	114.437353	41	41	

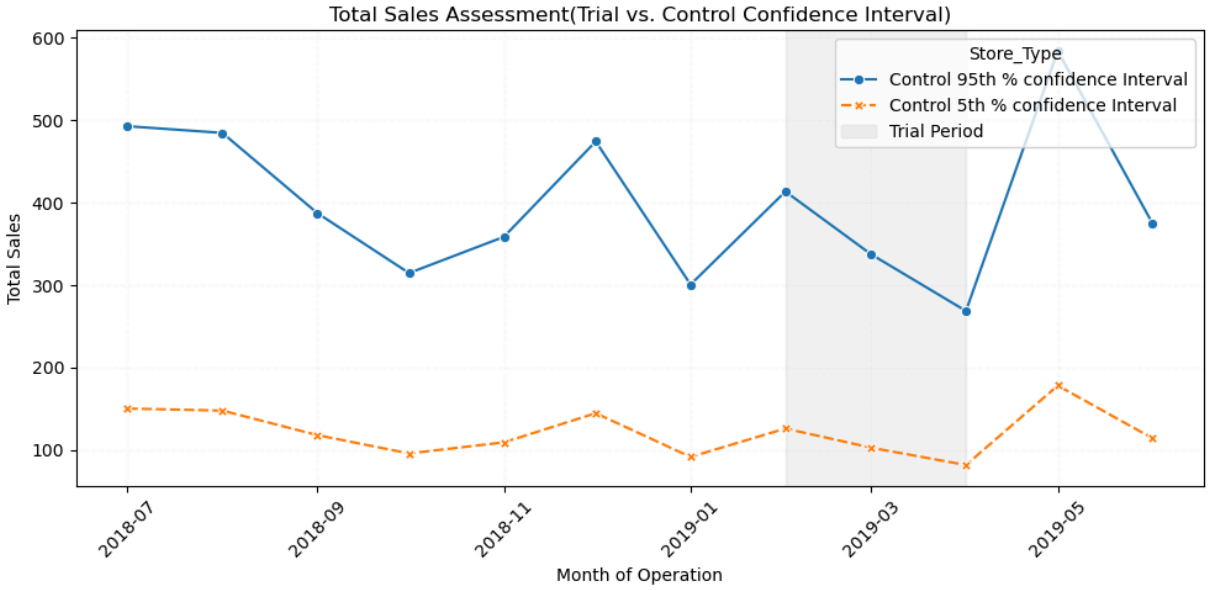
	chips_per_customer	avg_price_per_unit	scaledControlSales	\
0	1.058824	5.383333	295.311377	
1	1.041667	5.718000	290.435235	
2	1.071429	5.080000	232.226284	
3	1.028571	5.158333	188.645761	
4	1.025000	5.160976	214.956613	
5	1.063830	5.596000	284.238470	
6	1.000000	5.071429	180.315684	
7	1.044444	5.191489	247.870575	
8	1.025000	4.856098	202.258325	
9	1.100000	4.806061	161.115874	
10	1.087719	5.554839	349.863221	
11	1.000000	5.390244	224.505725	
12	1.058824	5.383333	295.311377	
13	1.041667	5.718000	290.435235	
14	1.071429	5.080000	232.226284	
15	1.028571	5.158333	188.645761	
16	1.025000	5.160976	214.956613	
17	1.063830	5.596000	284.238470	
18	1.000000	5.071429	180.315684	
19	1.044444	5.191489	247.870575	
20	1.025000	4.856098	202.258325	
21	1.100000	4.806061	161.115874	
22	1.087719	5.554839	349.863221	
23	1.000000	5.390244	224.505725	

	Store_Type	Transaction_Month
0	Control 95th % confidence Interval	2018-07-01
1	Control 95th % confidence Interval	2018-08-01
2	Control 95th % confidence Interval	2018-09-01

```
3   Control 95th % confidence Interval        2018-10-01
4   Control 95th % confidence Interval        2018-11-01
5   Control 95th % confidence Interval        2018-12-01
6   Control 95th % confidence Interval        2019-01-01
7   Control 95th % confidence Interval        2019-02-01
8   Control 95th % confidence Interval        2019-03-01
9   Control 95th % confidence Interval        2019-04-01
10  Control 95th % confidence Interval        2019-05-01
11  Control 95th % confidence Interval        2019-06-01
12   Control 5th % confidence Interval        2018-07-01
13   Control 5th % confidence Interval        2018-08-01
14   Control 5th % confidence Interval        2018-09-01
15   Control 5th % confidence Interval        2018-10-01
16   Control 5th % confidence Interval        2018-11-01
17   Control 5th % confidence Interval        2018-12-01
18   Control 5th % confidence Interval        2019-01-01
19   Control 5th % confidence Interval        2019-02-01
20   Control 5th % confidence Interval        2019-03-01
21   Control 5th % confidence Interval        2019-04-01
22   Control 5th % confidence Interval        2019-05-01
23   Control 5th % confidence Interval        2019-06-01
```

In [27]:

```python
def plot_assessment(trial_ass_df):
    plt.figure(figsize=(10,5))
    sns.lineplot(
    data=trial_ass_df,
    x='Transaction_Month',
    y='monthly_sales',
    hue='Store_Type',
    style='Store_Type',
    markers=True
    )
    plt.axvspan(
    pd.to_datetime('2019-02-01'),
    pd.to_datetime('2019-04-01'),
    color='grey',alpha=0.1,
    label='Trial Period'
    )
    plt.title("Total Sales Assessment(Trial vs. Control Confidence Interval)")
    plt.xlabel("Month of Operation")
    plt.ylabel("Total Sales")
    plt.grid(True,linestyle='--',alpha=0.06)
    plt.legend(title="Store_Type")
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
plot_assessment(trial_ass_df)
```



In [28]:
```
scaling_month=['201807','201808','201809','201810','201811','201812','201901','2019
def calculate_scaling_factor_cust(pre_trial,trial_store,control_store):
    """Calculates the scaling factor by comparing pre-trial sums."""
    #filter for the specific months
    df_scale=pre_trial[pre_trial['YEARMONTH'].astype(int)<=201902]
    #get trial stores sum for the scaling month
    trial_customers=df_scale[df_scale['STORE_NBR']==trial_store]['monthly_customers
    #get control store sales sum for scaling month
    control_customers=df_scale[df_scale['STORE_NBR']==control_store]['monthly_custo
    #Scaling_factor = Trial_sales/Control_sales
    if control_customers !=0:
        scaling_factor=trial_customers/control_customers
    else :
        scaling_factor=1.0
    return scaling_factor
scaling_factor_cust=calculate_scaling_factor_cust(pre_trial,77,233)
print(scaling_factor_cust)
```

```
1.0029154518950438
```

In [29]:
```
def apply_scaling_factor_cust(metrics,control_stores,scaling_factor):
    """Applies the scaling factore to the control store's 'monthly_sales' column.""
    #create a copy to avoid warning
    df_scaled_cust=metrics.copy()
    #apply scaling only to the control's data
    condition = (df_scaled_cust['STORE_NBR']==control_store)
    #calc the scaled sales clm
    df_scaled_cust.loc[condition,'scaledControlCust']= df_scaled_cust.loc[condition
    #for the trial store and other stores , set scaledControlStores to the original
    #we only care about the control store for this column
    df_scaled_cust.loc[~condition,'scaledControlCust']=df_scaled_cust.loc[~conditio
    return df_scaled_cust
metrics_scaledCust_df=apply_scaling_factor_cust(metrics,233,scaling_factor)
print(metrics_scaledCust_df)
```

```
      STORE_NBR YEARMONTH  monthly_sales  monthly_customers  transactions  \
0             1    201807          206.9                 49            52
1             1    201808          176.1                 42            43
2             1    201809          278.8                 59            62
3             1    201810          188.1                 44            45
4             1    201811          192.6                 46            47
...         ...       ...            ...                ...           ...
3164        272    201902          395.5                 45            48
3165        272    201903          442.3                 50            53
3166        272    201904          445.1                 54            56
3167        272    201905          314.6                 34            40
3168        272    201906          312.1                 34            37

      chips_per_customer  avg_price_per_unit  scaledControlCust
0               1.061224            3.978846               49.0
1               1.023810            4.095349               42.0
2               1.050847            4.496774               59.0
3               1.022727            4.180000               44.0
4               1.021739            4.097872               46.0
...                  ...                 ...                ...
3164            1.066667            8.239583               45.0
3165            1.060000            8.345283               50.0
3166            1.037037            7.948214               54.0
3167            1.176471            7.865000               34.0
3168            1.088235            8.435135               34.0

[3169 rows x 8 columns]
```

In [30]:
```
def calc_percentage_diff_cust(metrics_scaledCust_df,trial_store,control_store):
    """Calculate the percentage difference using filtering and merging"""
    trial_months = ['201902','201903','201904']
    control_data_cust=metrics_scaledCust_df[(metrics_scaledCust_df['STORE_NBR']==co
    (metrics_scaledCust_df['YEARMONTH'].isin(trial_months))].copy()
    control_data_cust= control_data_cust[['YEARMONTH','scaledControlCust']].rename(
    columns={
    'scaledControlCust':'ScaledControlCust'}
    )
    trial_data_cust = metrics_scaledCust_df[(metrics_scaledCust_df['STORE_NBR']==tr
    (metrics_scaledCust_df['YEARMONTH'].isin(trial_months))].copy()
    trial_data_cust= trial_data_cust[['YEARMONTH','monthly_customers']].rename(
    columns={
    'monthly_customers':'TrialCust_Actual'}
    )
    final_comparison_cust= pd.merge(control_data_cust,trial_data_cust,on='YEARMONTH
    final_comparison_cust['percentage_diff_cust']= (final_comparison_cust['TrialCus
    return final_comparison_cust
final_diff_table_cust=calc_percentage_diff_cust(metrics_scaledCust_df,77,233)
print(final_diff_table_cust)
```

```
  YEARMONTH  ScaledControlCust  TrialCust_Actual  percentage_diff_cust
0    201902          45.713835                45             -0.015615
1    201903          40.634520                50              0.230481
2    201904          30.475890                47              0.542203
```

In [31]:
```
final_diff_table_cust['YEARMONTH']=final_diff_table_cust['YEARMONTH'].astype(int)
stdDev_cust=final_diff_table_cust[final_diff_table_cust['YEARMONTH']>=201902]['perc
```

```
print(f"Standard Deviation of pre-trial percentage diff:{stdDev_cust:.4f}")
```

```
Standard Deviation of pre-trial percentage diff:0.2796
```

In [32]:
```
from scipy.stats import t
degreesOfFreedom=7 #8-1(trial period months)
trial_data2= final_diff_table_cust.copy()
#Calculate the t-value: t=(x-mu)/stdDev since mu(null hypothesis is 0:
#trial_data['tvalue']=trial_data['percentage_diff']/stdDev
trial_data2['tValue'] = np.where(stdDev_cust!=0,trial_data1['percentage_diff']/stdD
#find the 95th percentile of the t-distribution
t_percentile_cust_95=t.ppf(0.95,degreesOfFreedom)
print(f"(95th percentile t-value (df=6): {t_percentile_cust_95:.4f}")
print("\nTrial Month t-values:")
print(trial_data2[['YEARMONTH','percentage_diff_cust','tValue']])
```

```
(95th percentile t-value (df=6): 1.8946

Trial Month t-values:
   YEARMONTH  percentage_diff_cust    tValue
0     201902             -0.015615 -0.185742
1     201903              0.230481  1.348416
2     201904              0.542203  2.273172
```

In [33]:
```
def create_assessment_data_cust(metrics_scaledCust_df,stdDev_cust):
    """Calculates the 5th and 95th percentile sales band for the control stores."""
    control_store=233
    control_base_cust=metrics_scaledCust_df[metrics_scaledCust_df['STORE_NBR']== co
    #control 95th percentile (upper bound)
    pastCustControls95= control_base_cust.copy()
    #formula:totSales*(1+stdDev*2)
    pastCustControls95['monthly_customers']=pastCustControls95['monthly_customers']
    pastCustControls95['Store_Type']="Control 95th % confidence Interval"
    #control 5th percentile (Lower Bound)
    pastCustControls5= control_base_cust.copy()
    pastCustControls5['monthly_customers']=pastCustControls95['monthly_customers']*
    pastCustControls5['Store_Type']="Control 5th % confidence Interval"
    trial_assessment1= pd.concat([pastCustControls95,pastCustControls5],ignore_inde
    trial_assessment1['Transaction_Month']=pd.to_datetime(
        trial_assessment1['YEARMONTH'].astype(str),format='%Y%m')
    return trial_assessment1
trial_ass_cust_df= create_assessment_data_cust(metrics_scaledCust_df,stdDev_cust)
print(trial_ass_cust_df)
```

	STORE_NBR	YEARMONTH	monthly_sales	monthly_customers	transactions	\
0	233	201807	290.7	79.514268	54	
1	233	201808	285.9	74.836958	50	
2	233	201809	228.6	65.482338	45	
3	233	201810	185.7	54.568615	36	
4	233	201811	211.6	62.364132	41	
5	233	201812	279.8	73.277855	50	
6	233	201901	177.5	54.568615	35	
7	233	201902	244.0	70.159648	47	
8	233	201903	199.1	62.364132	41	
9	233	201904	158.6	46.773099	33	
10	233	201905	344.4	88.868888	62	
11	233	201906	221.0	63.923235	41	
12	233	201807	290.7	35.057579	54	
13	233	201808	285.9	32.995369	50	
14	233	201809	228.6	28.870947	45	
15	233	201810	185.7	24.059123	36	
16	233	201811	211.6	27.496140	41	
17	233	201812	279.8	32.307965	50	
18	233	201901	177.5	24.059123	35	
19	233	201902	244.0	30.933158	47	
20	233	201903	199.1	27.496140	41	
21	233	201904	158.6	20.622105	33	
22	233	201905	344.4	39.182000	62	
23	233	201906	221.0	28.183544	41	

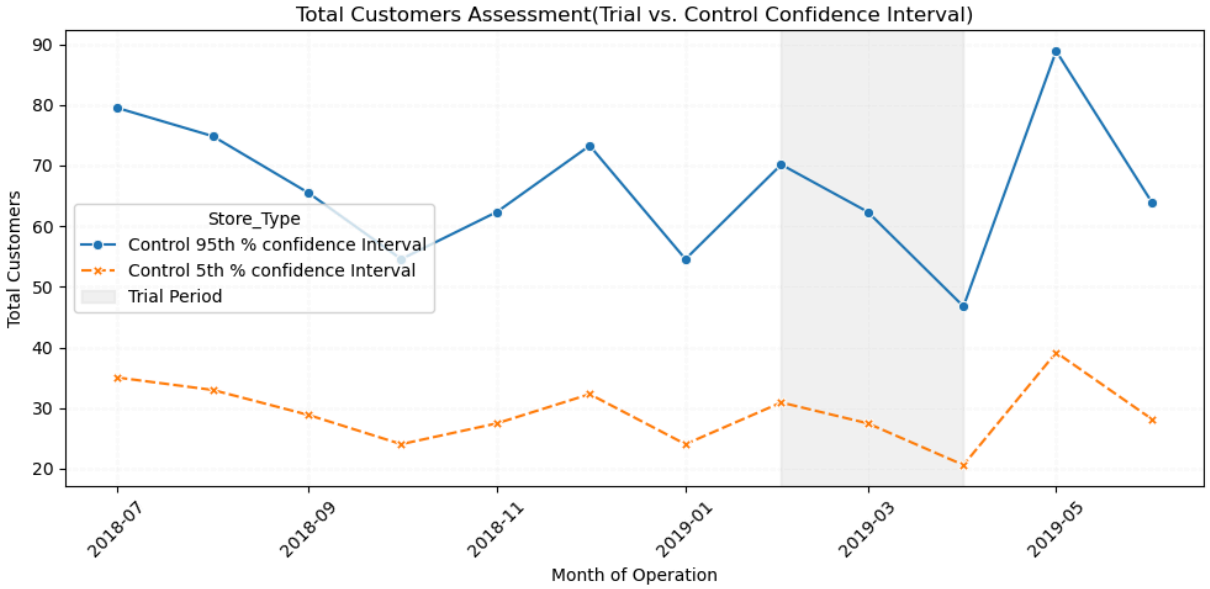
	chips_per_customer	avg_price_per_unit	scaledControlCust	\
0	1.058824	5.383333	51.809014	
1	1.041667	5.718000	48.761425	
2	1.071429	5.080000	42.666246	
3	1.028571	5.158333	35.555205	
4	1.025000	5.160976	40.634520	
5	1.063830	5.596000	47.745562	
6	1.000000	5.071429	35.555205	
7	1.044444	5.191489	45.713835	
8	1.025000	4.856098	40.634520	
9	1.100000	4.806061	30.475890	
10	1.087719	5.554839	57.904192	
11	1.000000	5.390244	41.650383	
12	1.058824	5.383333	51.809014	
13	1.041667	5.718000	48.761425	
14	1.071429	5.080000	42.666246	
15	1.028571	5.158333	35.555205	
16	1.025000	5.160976	40.634520	
17	1.063830	5.596000	47.745562	
18	1.000000	5.071429	35.555205	
19	1.044444	5.191489	45.713835	
20	1.025000	4.856098	40.634520	
21	1.100000	4.806061	30.475890	
22	1.087719	5.554839	57.904192	
23	1.000000	5.390244	41.650383	

	Store_Type	Transaction_Month
0	Control 95th % confidence Interval	2018-07-01
1	Control 95th % confidence Interval	2018-08-01
2	Control 95th % confidence Interval	2018-09-01

```
3   Control 95th % confidence Interval        2018-10-01
4   Control 95th % confidence Interval        2018-11-01
5   Control 95th % confidence Interval        2018-12-01
6   Control 95th % confidence Interval        2019-01-01
7   Control 95th % confidence Interval        2019-02-01
8   Control 95th % confidence Interval        2019-03-01
9   Control 95th % confidence Interval        2019-04-01
10  Control 95th % confidence Interval        2019-05-01
11  Control 95th % confidence Interval        2019-06-01
12   Control 5th % confidence Interval        2018-07-01
13   Control 5th % confidence Interval        2018-08-01
14   Control 5th % confidence Interval        2018-09-01
15   Control 5th % confidence Interval        2018-10-01
16   Control 5th % confidence Interval        2018-11-01
17   Control 5th % confidence Interval        2018-12-01
18   Control 5th % confidence Interval        2019-01-01
19   Control 5th % confidence Interval        2019-02-01
20   Control 5th % confidence Interval        2019-03-01
21   Control 5th % confidence Interval        2019-04-01
22   Control 5th % confidence Interval        2019-05-01
23   Control 5th % confidence Interval        2019-06-01
```

In [34]:

```python
def plot_assessment_cust(trial_ass_cust_df):
    plt.figure(figsize=(10,5))
    sns.lineplot(
    data=trial_ass_cust_df,
    x='Transaction_Month',
    y='monthly_customers',
    hue='Store_Type',
    style='Store_Type',
    markers=True
    )
    plt.axvspan(
    pd.to_datetime('2019-02-01'),
    pd.to_datetime('2019-04-01'),
    color='grey',alpha=0.1,
    label='Trial Period'
    )
    plt.title("Total Customers Assessment(Trial vs. Control Confidence Interval)")
    plt.xlabel("Month of Operation")
    plt.ylabel("Total Customers")
    plt.grid(True,linestyle='--',alpha=0.06)
    plt.legend(title="Store_Type")
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
plot_assessment_cust(trial_ass_cust_df)
```



In [35]:
```
#Previously we were evaluating for the trial score = 77, now we will evaluate for t
```

In [36]:
```
trial_store_list=[86]
sales_ranking86=calc_correlation(pre_trial,'monthly_sales',trial_store_list)
print(sales_ranking86)
customer_ranking86=calc_correlation(pre_trial,'monthly_customers',trial_store_list)
print(customer_ranking86)
```

```
     STORE_NBR  Correlation  Trial_Store
0          155     0.841589           86
1          260     0.736378           86
2           22     0.722506           86
3            6     0.683779           86
4          269     0.681775           86
..         ...          ...          ...
254        185    -0.776159           86
255        254    -0.780106           86
256        214    -0.783369           86
257        256    -0.849045           86
258        120    -0.876296           86

[259 rows x 3 columns]
     STORE_NBR  Correlation  Trial_Store
0          260     0.772289           86
1          147     0.728606           86
2          176     0.710471           86
3           74     0.701439           86
4          155     0.646118           86
..         ...          ...          ...
254         27    -0.785631           86
255        270    -0.803394           86
256        120    -0.814822           86
257         48    -0.815227           86
258        259    -0.877478           86

[259 rows x 3 columns]
```

In [37]:
```
sales_mag_ranking86= calc_magnitude_distance(pre_trial,'monthly_sales',trial_store_
print(sales_mag_ranking86.sort_values(by='Magnitude_Measure',ascending=False).head(
```

```
     Trial_Store  Control_Store  Magnitude_Measure
146           86            155           1.000000
101           86            109           0.995972
212           86            225           0.978371
216           86            229           0.976342
9             86             10           0.976049
```

In [38]:
```
customer_mag_ranking86= calc_magnitude_distance(pre_trial,'monthly_customers',trial
print(customer_mag_ranking86.sort_values(by='Magnitude_Measure',ascending=False).he
```

```
     Trial_Store  Control_Store  Magnitude_Measure
146           86            155           1.000000
216           86            229           0.984085
53            86             57           0.982759
101           86            109           0.977454
212           86            225           0.976127
```

In [39]:
```
score_sales86=pd.merge(sales_ranking86,sales_mag_ranking86,on=['Trial_Store'],suffi
#calculate combined sales score : 0.5*Correlation + 0.5*Magnitude_measure
corr_weight = 0.5
mag_weight = 1-corr_weight
score_sales86['scoreSales']=(score_sales86['Correlation']*corr_weight)+(score_sales
print(score_sales86)
```

```
       STORE_NBR  Correlation  Trial_Store  Control_Store  Magnitude_Measure  \
0            155     0.841589           86              1           0.222693
1            155     0.841589           86              2           0.170928
2            155     0.841589           86              3           0.783327
3            155     0.841589           86              4           0.580992
4            155     0.841589           86              5           0.940915
...          ...          ...          ...            ...                ...
67076        120    -0.876296           86            268           0.237987
67077        120    -0.876296           86            269           0.940689
67078        120    -0.876296           86            270           0.883355
67079        120    -0.876296           86            271           0.939382
67080        120    -0.876296           86            272           0.452293

       scoreSales
0        0.532141
1        0.506259
2        0.812458
3        0.711291
4        0.891252
...           ...
67076   -0.319154
67077    0.032197
67078    0.003530
67079    0.031543
67080   -0.212001

[67081 rows x 6 columns]
```

In [40]:
```
score_customer86=pd.merge(customer_ranking86,customer_mag_ranking86,on=['Trial_Stor
#calculate combined sales score : 0.5*Correlation + 0.5*Magnitude_measure
```

```
corr_weight = 0.5
mag_weight = 1-corr_weight
score_customer86['scoreNcust']=(score_customer86['Correlation']*corr_weight)+(score
print(score_customer86)
```

```
       STORE_NBR  Correlation  Trial_Store  Control_Store  Magnitude_Measure  \
0            260     0.772289           86              1           0.452255
1            260     0.772289           86              2           0.362069
2            260     0.772289           86              3           0.938992
3            260     0.772289           86              4           0.806366
4            260     0.772289           86              5           0.933687
...          ...          ...          ...            ...                ...
67076        259    -0.877478           86            268           0.413793
67077        259    -0.877478           86            269           0.950928
67078        259    -0.877478           86            270           0.915119
67079        259    -0.877478           86            271           0.941645
67080        259    -0.877478           86            272           0.423077

       scoreNcust
0        0.612272
1        0.567179
2        0.855640
3        0.789327
4        0.852988
...           ...
67076   -0.231842
67077    0.036725
67078    0.018821
67079    0.032083
67080   -0.227200

[67081 rows x 6 columns]
```

In [41]:
```
final_score_ranking86=pd.merge(score_sales86,score_customer86,on=['Control_Store','
print(final_score_ranking86)
```

```
          STORE_NBR_sales  Correlation_sales  Trial_Store  Control_Store  \
0                     155           0.841589           86              1
1                     155           0.841589           86              1
2                     155           0.841589           86              1
3                     155           0.841589           86              1
4                     155           0.841589           86              1
...                   ...                ...          ...            ...
17373974              120          -0.876296           86            272
17373975              120          -0.876296           86            272
17373976              120          -0.876296           86            272
17373977              120          -0.876296           86            272
17373978              120          -0.876296           86            272

          Magnitude_Measure_sales  scoreSales  STORE_NBR_cust  \
0                        0.222693    0.532141             260
1                        0.222693    0.532141             147
2                        0.222693    0.532141             176
3                        0.222693    0.532141              74
4                        0.222693    0.532141             155
...                           ...         ...             ...
17373974                 0.452293   -0.212001              27
17373975                 0.452293   -0.212001             270
17373976                 0.452293   -0.212001             120
17373977                 0.452293   -0.212001              48
17373978                 0.452293   -0.212001             259

          Correlation_cust  Magnitude_Measure_cust  scoreNcust
0                 0.772289                0.452255    0.612272
1                 0.728606                0.452255    0.590430
2                 0.710471                0.452255    0.581363
3                 0.701439                0.452255    0.576847
4                 0.646118                0.452255    0.549187
...                    ...                     ...         ...
17373974         -0.785631                0.423077   -0.181277
17373975         -0.803394                0.423077   -0.190159
17373976         -0.814822                0.423077   -0.195873
17373977         -0.815227                0.423077   -0.196075
17373978         -0.877478                0.423077   -0.227200

[17373979 rows x 10 columns]
```

In [42]:
```
#calculate the final composite score using the suffixed column names
final_score_ranking86['finalControlScore']=(final_score_ranking86['scoreSales']*0.5
final_ranking86=final_score_ranking86.sort_values(by=['Trial_Store','finalControlSc
print(final_ranking86[['Trial_Store','Control_Store','finalControlScore']].head(10)
```

```
   Trial_Store  Control_Store  finalControlScore
0           86            155           0.903469
1           86            109           0.896826
2           86            229           0.893576
3           86            155           0.892549
4           86            225           0.892094
5           86            247           0.889717
6           86            155           0.888015
7           86             62           0.887891
8           86             57           0.887522
9           86             10           0.886208
```

In [43]:
```
scaling_factor_86=calculate_scaling_factor(pre_trial,86,155)
print(scaling_factor_86)
```

```
0.9768260670287088
```

In [44]:
```
scaled_df_86 = apply_scaling_factor(metrics,155,scaling_factor_86)
print(scaled_df_86)
```

```
      STORE_NBR YEARMONTH  monthly_sales  monthly_customers  transactions  \
0             1    201807          206.9                 49            52
1             1    201808          176.1                 42            43
2             1    201809          278.8                 59            62
3             1    201810          188.1                 44            45
4             1    201811          192.6                 46            47
...         ...       ...            ...                ...           ...
3164        272    201902          395.5                 45            48
3165        272    201903          442.3                 50            53
3166        272    201904          445.1                 54            56
3167        272    201905          314.6                 34            40
3168        272    201906          312.1                 34            37

      chips_per_customer  avg_price_per_unit  scaledControlSales
0               1.061224            3.978846               206.9
1               1.023810            4.095349               176.1
2               1.050847            4.496774               278.8
3               1.022727            4.180000               188.1
4               1.021739            4.097872               192.6
...                  ...                 ...                 ...
3164            1.066667            8.239583               395.5
3165            1.060000            8.345283               442.3
3166            1.037037            7.948214               445.1
3167            1.176471            7.865000               314.6
3168            1.088235            8.435135               312.1

[3169 rows x 8 columns]
```

In [138…
```
final_diff_table_86=calc_percentage_diff(metrics_scaled_df,86,155)
print(final_diff_table_86)
```

```
  YEARMONTH  ScaledControlSales  TrialSales_Actual  percentage_diff
0    201902               891.2              913.2         0.024686
1    201903               804.4             1026.8         0.276479
2    201904               844.6              848.2         0.004262
```

In [140…
```
final_diff_table_86['YEARMONTH']=final_diff_table_86['YEARMONTH'].astype(int)
stdDev_86=final_diff_table_86[final_diff_table_86['YEARMONTH']>=201902]['percentage
print(f"Standard Deviation of pre-trial percentage diff:{stdDev_86:.4f}")
```

```
Standard Deviation of pre-trial percentage diff:0.1516
```

In [152…
```
from scipy.stats import t
degreesOfFreedom=7 #8-1(trial period months)
trial_data_86= final_diff_table_86.copy()
#Calculate the t-value: t=(x-mu)/stdDev since mu(null hypothesis is 0:
#trial_data['tvalue']=trial_data['percentage_diff']/stdDev
trial_data_86['tValue'] = np.where(stdDev_86!=0,trial_data_86['percentage_diff']/st
#find the 95th percentile of the t-distribution
t_percentile_95=t.ppf(0.95,degreesOfFreedom)
print(f"(95th percentile t-value (df=7): {t_percentile_95:.4f}")
print("\nTrial Month t-values:")
print(trial_data_86[['YEARMONTH','percentage_diff','tValue']])
```

```
(95th percentile t-value (df=7): 1.8946

Trial Month t-values:
   YEARMONTH  percentage_diff    tValue
0     201902         0.024686  0.162821
1     201903         0.276479  1.823585
2     201904         0.004262  0.028113
```

In [48]:
```
trial_ass86_df= create_assessment_data(metrics_scaled_df,stdDev_86)
print(trial_ass86_df)
```

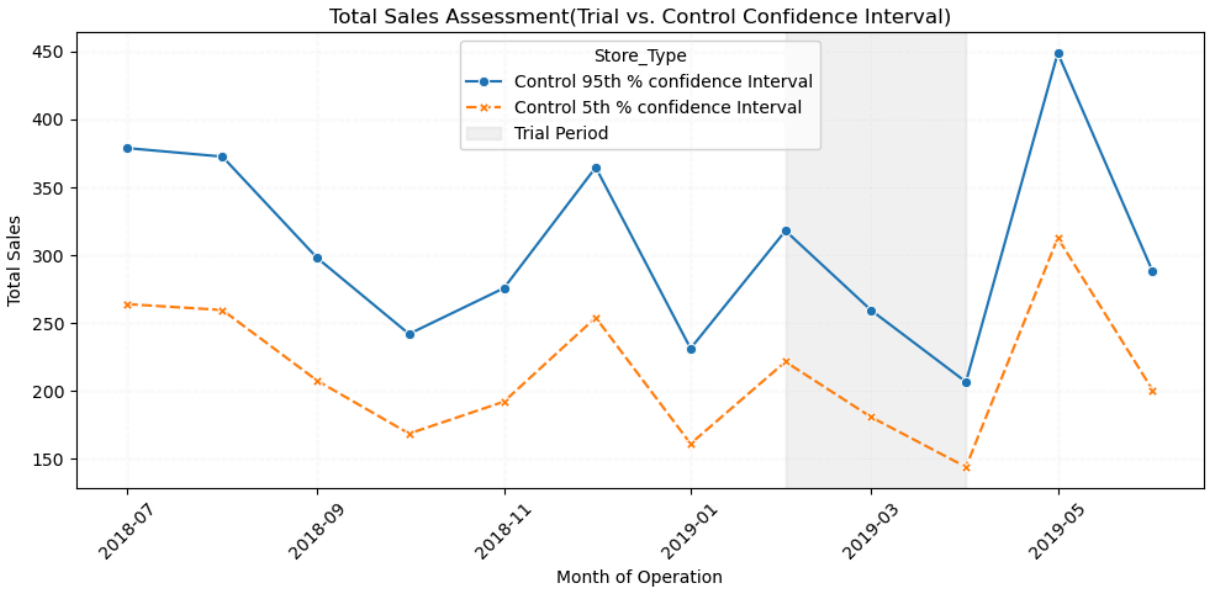
	STORE_NBR	YEARMONTH	monthly_sales	monthly_customers	transactions	\
0	233	201807	378.847858	51	54	
1	233	201808	372.592372	48	50	
2	233	201809	297.917511	42	45	
3	233	201810	242.009106	35	36	
4	233	201811	275.762665	40	41	
5	233	201812	364.642692	47	50	
6	233	201901	231.322651	35	35	
7	233	201902	317.987194	45	47	
8	233	201903	259.472337	40	41	
9	233	201904	206.691676	30	33	
10	233	201905	448.831105	57	62	
11	233	201906	288.012991	41	41	
12	233	201807	263.971260	51	54	
13	233	201808	259.612601	48	50	
14	233	201809	207.581114	42	45	
15	233	201810	168.625603	35	36	
16	233	201811	192.144199	40	41	
17	233	201812	254.073472	47	50	
18	233	201901	161.179562	35	35	
19	233	201902	221.565144	45	47	
20	233	201903	180.793525	40	41	
21	233	201904	144.017344	30	33	
22	233	201905	312.733752	57	62	
23	233	201906	200.679905	41	41	

	chips_per_customer	avg_price_per_unit	scaledControlSales	\
0	1.058824	5.383333	295.311377	
1	1.041667	5.718000	290.435235	
2	1.071429	5.080000	232.226284	
3	1.028571	5.158333	188.645761	
4	1.025000	5.160976	214.956613	
5	1.063830	5.596000	284.238470	
6	1.000000	5.071429	180.315684	
7	1.044444	5.191489	247.870575	
8	1.025000	4.856098	202.258325	
9	1.100000	4.806061	161.115874	
10	1.087719	5.554839	349.863221	
11	1.000000	5.390244	224.505725	
12	1.058824	5.383333	295.311377	
13	1.041667	5.718000	290.435235	
14	1.071429	5.080000	232.226284	
15	1.028571	5.158333	188.645761	
16	1.025000	5.160976	214.956613	
17	1.063830	5.596000	284.238470	
18	1.000000	5.071429	180.315684	
19	1.044444	5.191489	247.870575	
20	1.025000	4.856098	202.258325	
21	1.100000	4.806061	161.115874	
22	1.087719	5.554839	349.863221	
23	1.000000	5.390244	224.505725	

	Store_Type	Transaction_Month
0	Control 95th % confidence Interval	2018-07-01
1	Control 95th % confidence Interval	2018-08-01
2	Control 95th % confidence Interval	2018-09-01

```
3   Control 95th % confidence Interval        2018-10-01
4   Control 95th % confidence Interval        2018-11-01
5   Control 95th % confidence Interval        2018-12-01
6   Control 95th % confidence Interval        2019-01-01
7   Control 95th % confidence Interval        2019-02-01
8   Control 95th % confidence Interval        2019-03-01
9   Control 95th % confidence Interval        2019-04-01
10  Control 95th % confidence Interval        2019-05-01
11  Control 95th % confidence Interval        2019-06-01
12   Control 5th % confidence Interval        2018-07-01
13   Control 5th % confidence Interval        2018-08-01
14   Control 5th % confidence Interval        2018-09-01
15   Control 5th % confidence Interval        2018-10-01
16   Control 5th % confidence Interval        2018-11-01
17   Control 5th % confidence Interval        2018-12-01
18   Control 5th % confidence Interval        2019-01-01
19   Control 5th % confidence Interval        2019-02-01
20   Control 5th % confidence Interval        2019-03-01
21   Control 5th % confidence Interval        2019-04-01
22   Control 5th % confidence Interval        2019-05-01
23   Control 5th % confidence Interval        2019-06-01
```

In [49]:
```
plot_assessment(trial_ass86_df)
```



In [50]:
```
scaling_factor86_cust=calculate_scaling_factor_cust(pre_trial,86,155)
print(scaling_factor86_cust)
```

```
1.0151515151515151
```

In [51]:
```
metrics_scaledCust86_df=apply_scaling_factor_cust(metrics,155,scaling_factor86_cust
print(metrics_scaledCust86_df)
```

```
      STORE_NBR YEARMONTH  monthly_sales  monthly_customers  transactions  \
0             1    201807          206.9                 49            52
1             1    201808          176.1                 42            43
2             1    201809          278.8                 59            62
3             1    201810          188.1                 44            45
4             1    201811          192.6                 46            47
...         ...       ...            ...                ...           ...
3164        272    201902          395.5                 45            48
3165        272    201903          442.3                 50            53
3166        272    201904          445.1                 54            56
3167        272    201905          314.6                 34            40
3168        272    201906          312.1                 34            37

      chips_per_customer  avg_price_per_unit  scaledControlCust
0               1.061224            3.978846               49.0
1               1.023810            4.095349               42.0
2               1.050847            4.496774               59.0
3               1.022727            4.180000               44.0
4               1.021739            4.097872               46.0
...                  ...                 ...                ...
3164            1.066667            8.239583               45.0
3165            1.060000            8.345283               50.0
3166            1.037037            7.948214               54.0
3167            1.176471            7.865000               34.0
3168            1.088235            8.435135               34.0

[3169 rows x 8 columns]
```

In [52]:
```
final_diff_table86_cust=calc_percentage_diff_cust(metrics_scaledCust86_df,86,155)
print(final_diff_table86_cust)
```

```
  YEARMONTH  ScaledControlCust  TrialCust_Actual  percentage_diff_cust
0    201902               95.0               107              0.126316
1    201903               94.0               115              0.223404
2    201904               99.0               105              0.060606
```

In [53]:
```
final_diff_table86_cust['YEARMONTH']=final_diff_table86_cust['YEARMONTH'].astype(in
stdDev86_cust=final_diff_table86_cust[final_diff_table86_cust['YEARMONTH']>=201902]
print(f"Standard Deviation of pre-trial percentage diff:{stdDev86_cust:.4f}")
```

```
Standard Deviation of pre-trial percentage diff:0.0819
```

In [54]:
```
trial_ass_cust86_df= create_assessment_data_cust(metrics_scaledCust86_df,stdDev86_c
print(trial_ass_cust86_df)
```

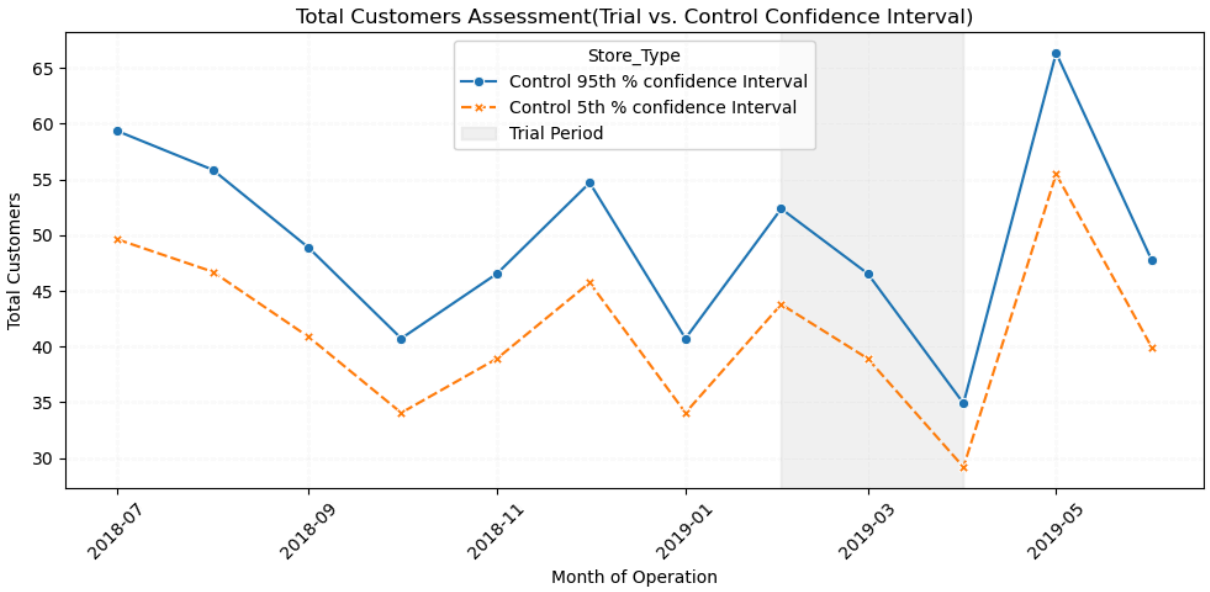
	STORE_NBR	YEARMONTH	monthly_sales	monthly_customers	transactions	\
0	233	201807	290.7	59.353959	54	
1	233	201808	285.9	55.862550	50	
2	233	201809	228.6	48.879731	45	
3	233	201810	185.7	40.733109	36	
4	233	201811	211.6	46.552125	41	
5	233	201812	279.8	54.698746	50	
6	233	201901	177.5	40.733109	35	
7	233	201902	244.0	52.371140	47	
8	233	201903	199.1	46.552125	41	
9	233	201904	158.6	34.914093	33	
10	233	201905	344.4	66.336778	62	
11	233	201906	221.0	47.715928	41	
12	233	201807	290.7	49.631596	54	
13	233	201808	285.9	46.712090	50	
14	233	201809	228.6	40.873079	45	
15	233	201810	185.7	34.060899	36	
16	233	201811	211.6	38.926742	41	
17	233	201812	279.8	45.738921	50	
18	233	201901	177.5	34.060899	35	
19	233	201902	244.0	43.792584	47	
20	233	201903	199.1	38.926742	41	
21	233	201904	158.6	29.195056	33	
22	233	201905	344.4	55.470607	62	
23	233	201906	221.0	39.899910	41	

	chips_per_customer	avg_price_per_unit	scaledControlCust	\
0	1.058824	5.383333	51.772727	
1	1.041667	5.718000	48.727273	
2	1.071429	5.080000	42.636364	
3	1.028571	5.158333	35.530303	
4	1.025000	5.160976	40.606061	
5	1.063830	5.596000	47.712121	
6	1.000000	5.071429	35.530303	
7	1.044444	5.191489	45.681818	
8	1.025000	4.856098	40.606061	
9	1.100000	4.806061	30.454545	
10	1.087719	5.554839	57.863636	
11	1.000000	5.390244	41.621212	
12	1.058824	5.383333	51.772727	
13	1.041667	5.718000	48.727273	
14	1.071429	5.080000	42.636364	
15	1.028571	5.158333	35.530303	
16	1.025000	5.160976	40.606061	
17	1.063830	5.596000	47.712121	
18	1.000000	5.071429	35.530303	
19	1.044444	5.191489	45.681818	
20	1.025000	4.856098	40.606061	
21	1.100000	4.806061	30.454545	
22	1.087719	5.554839	57.863636	
23	1.000000	5.390244	41.621212	

	Store_Type	Transaction_Month
0	Control 95th % confidence Interval	2018-07-01
1	Control 95th % confidence Interval	2018-08-01
2	Control 95th % confidence Interval	2018-09-01

```
3   Control 95th % confidence Interval        2018-10-01
4   Control 95th % confidence Interval        2018-11-01
5   Control 95th % confidence Interval        2018-12-01
6   Control 95th % confidence Interval        2019-01-01
7   Control 95th % confidence Interval        2019-02-01
8   Control 95th % confidence Interval        2019-03-01
9   Control 95th % confidence Interval        2019-04-01
10  Control 95th % confidence Interval        2019-05-01
11  Control 95th % confidence Interval        2019-06-01
12   Control 5th % confidence Interval        2018-07-01
13   Control 5th % confidence Interval        2018-08-01
14   Control 5th % confidence Interval        2018-09-01
15   Control 5th % confidence Interval        2018-10-01
16   Control 5th % confidence Interval        2018-11-01
17   Control 5th % confidence Interval        2018-12-01
18   Control 5th % confidence Interval        2019-01-01
19   Control 5th % confidence Interval        2019-02-01
20   Control 5th % confidence Interval        2019-03-01
21   Control 5th % confidence Interval        2019-04-01
22   Control 5th % confidence Interval        2019-05-01
23   Control 5th % confidence Interval        2019-06-01
```

In [55]:
```
plot_assessment_cust(trial_ass_cust86_df)
```



In [114…
```
#Trial Store = 88
```

In [116…
```
trial_store_list=[88]
sales_ranking88=calc_correlation(pre_trial,'monthly_sales',trial_store_list)
print(sales_ranking88)
customer_ranking88=calc_correlation(pre_trial,'monthly_customers',trial_store_list)
print(customer_ranking88)
```

```
     STORE_NBR  Correlation  Trial_Store
0          159     0.895637           88
1            1     0.823306           88
2           91     0.783157           88
3          188     0.731655           88
4           61     0.726877           88
..         ...          ...          ...
254        270    -0.737748           88
255        272    -0.747878           88
256         48    -0.769634           88
257         23    -0.785574           88
258          8    -0.825262           88

[259 rows x 3 columns]
     STORE_NBR  Correlation  Trial_Store
0          237     0.942232           88
1           14     0.928985           88
2           35     0.901397           88
3          178     0.873835           88
4          265     0.842573           88
..         ...          ...          ...
254        147    -0.629198           88
255         19    -0.635319           88
256        239    -0.666731           88
257        247    -0.792942           88
258        258    -0.827778           88

[259 rows x 3 columns]
```

In [118…
```
customer_mag_ranking88= calc_magnitude_distance(pre_trial,'monthly_customers',trial
print(customer_mag_ranking88.sort_values(by='Magnitude_Measure',ascending=False).he
```

```
     Trial_Store  Control_Store  Magnitude_Measure
224           88            237           1.000000
37            88             40           0.958463
193           88            203           0.951194
189           88            199           0.948079
156           88            165           0.941848
```

In [122…
```
sales_mag_ranking88= calc_magnitude_distance(pre_trial,'monthly_sales',trial_store_
print(sales_mag_ranking88.sort_values(by='Magnitude_Measure',ascending=False).head(
```

```
     Trial_Store  Control_Store  Magnitude_Measure
224           88            237           1.000000
37            88             40           0.985029
193           88            203           0.973840
189           88            199           0.958269
156           88            165           0.952797
```

In [124…
```
score_sales88=pd.merge(sales_ranking88,sales_mag_ranking88,on=['Trial_Store'],suffi
#calculate combined sales score : 0.5*Correlation + 0.5*Magnitude_measure
corr_weight = 0.5
mag_weight = 1-corr_weight
score_sales88['scoreSales']=(score_sales88['Correlation']*corr_weight)+(score_sales
print(score_sales88)
```

```
       STORE_NBR  Correlation  Trial_Store  Control_Store  Magnitude_Measure  \
0            159     0.895637           88              1           0.145547
1            159     0.895637           88              2           0.111715
2            159     0.895637           88              3           0.844162
3            159     0.895637           88              4           0.902078
4            159     0.895637           88              5           0.622427
...          ...          ...          ...            ...                ...
67076          8    -0.825262           88            268           0.155543
67077          8    -0.825262           88            269           0.735694
67078          8    -0.825262           88            270           0.730453
67079          8    -0.825262           88            271           0.626809
67080          8    -0.825262           88            272           0.295609

       scoreSales
0        0.520592
1        0.503676
2        0.869899
3        0.898857
4        0.759032
...           ...
67076   -0.334859
67077   -0.044784
67078   -0.047404
67079   -0.099227
67080   -0.264826

[67081 rows x 6 columns]
```

In [126…

```
score_customer88=pd.merge(customer_ranking88,customer_mag_ranking88,on=['Trial_Stor
#calculate combined sales score : 0.5*Correlation + 0.5*Magnitude_measure
corr_weight = 0.5
mag_weight = 1-corr_weight
score_customer88['scoreNcust']=(score_customer88['Correlation']*corr_weight)+(score
print(score_customer88)
```

```
       STORE_NBR  Correlation  Trial_Store  Control_Store  Magnitude_Measure  \
0            237     0.942232           88              1           0.354102
1            237     0.942232           88              2           0.283489
2            237     0.942232           88              3           0.863967
3            237     0.942232           88              4           0.911734
4            237     0.942232           88              5           0.737279
...          ...          ...          ...            ...                ...
67076        258    -0.827778           88            268           0.323988
67077        258    -0.827778           88            269           0.852544
67078        258    -0.827778           88            270           0.830737
67079        258    -0.827778           88            271           0.737279
67080        258    -0.827778           88            272           0.331256

       scoreNcust
0        0.648167
1        0.612860
2        0.903099
3        0.926983
4        0.839756
...           ...
67076   -0.251895
67077    0.012383
67078    0.001479
67079   -0.045249
67080   -0.248261

[67081 rows x 6 columns]
```

In [128…

```
final_score_ranking88=pd.merge(score_sales88,score_customer88,on=['Control_Store','
print(final_score_ranking88)
```

```
          STORE_NBR_sales  Correlation_sales  Trial_Store  Control_Store  \
0                     159           0.895637           88              1
1                     159           0.895637           88              1
2                     159           0.895637           88              1
3                     159           0.895637           88              1
4                     159           0.895637           88              1
...                   ...                ...          ...            ...
17373974                8          -0.825262           88            272
17373975                8          -0.825262           88            272
17373976                8          -0.825262           88            272
17373977                8          -0.825262           88            272
17373978                8          -0.825262           88            272

          Magnitude_Measure_sales  scoreSales  STORE_NBR_cust  \
0                        0.145547    0.520592             237
1                        0.145547    0.520592              14
2                        0.145547    0.520592              35
3                        0.145547    0.520592             178
4                        0.145547    0.520592             265
...                           ...         ...             ...
17373974                 0.295609   -0.264826             147
17373975                 0.295609   -0.264826              19
17373976                 0.295609   -0.264826             239
17373977                 0.295609   -0.264826             247
17373978                 0.295609   -0.264826             258

          Correlation_cust  Magnitude_Measure_cust  scoreNcust
0                 0.942232                0.354102    0.648167
1                 0.928985                0.354102    0.641544
2                 0.901397                0.354102    0.627749
3                 0.873835                0.354102    0.613969
4                 0.842573                0.354102    0.598337
...                    ...                     ...         ...
17373974         -0.629198                0.331256   -0.148971
17373975         -0.635319                0.331256   -0.152031
17373976         -0.666731                0.331256   -0.167737
17373977         -0.792942                0.331256   -0.230843
17373978         -0.827778                0.331256   -0.248261

[17373979 rows x 10 columns]
```

In [130…

```
#calculate the final composite score using the suffixed column names
final_score_ranking88['finalControlScore']=(final_score_ranking88['scoreSales']*0.5
final_ranking88=final_score_ranking88.sort_values(by=['Trial_Store','finalControlSc
print(final_ranking88[['Trial_Store','Control_Store','finalControlScore']].head(10)
```

```
   Trial_Store  Control_Store  finalControlScore
0           88            237           0.959467
1           88            237           0.956155
2           88            237           0.949258
3           88             40           0.945340
4           88            237           0.942368
5           88             40           0.942028
6           88            237           0.941385
7           88            203           0.940726
8           88            237           0.938073
9           88            203           0.937414
```

In [132…
```
scaling_factor_88=calculate_scaling_factor(pre_trial,88,237)
print(scaling_factor_88)
```

```
0.998143644767863
```

In [134…
```
scaled_df_88 = apply_scaling_factor(metrics,237,scaling_factor_86)
print(scaled_df_88)
```

```
      STORE_NBR YEARMONTH  monthly_sales  monthly_customers  transactions  \
0             1    201807          206.9                 49            52
1             1    201808          176.1                 42            43
2             1    201809          278.8                 59            62
3             1    201810          188.1                 44            45
4             1    201811          192.6                 46            47
...         ...       ...            ...                ...           ...
3164        272    201902          395.5                 45            48
3165        272    201903          442.3                 50            53
3166        272    201904          445.1                 54            56
3167        272    201905          314.6                 34            40
3168        272    201906          312.1                 34            37

      chips_per_customer  avg_price_per_unit  scaledControlSales
0               1.061224            3.978846               206.9
1               1.023810            4.095349               176.1
2               1.050847            4.496774               278.8
3               1.022727            4.180000               188.1
4               1.021739            4.097872               192.6
...                  ...                 ...                 ...
3164            1.066667            8.239583               395.5
3165            1.060000            8.345283               442.3
3166            1.037037            7.948214               445.1
3167            1.176471            7.865000               314.6
3168            1.088235            8.435135               312.1

[3169 rows x 8 columns]
```

In [136…
```
final_diff_table_88=calc_percentage_diff(metrics_scaled_df,88,237)
print(final_diff_table_88)
```

```
  YEARMONTH  ScaledControlSales  TrialSales_Actual  percentage_diff
0    201902              1404.8             1370.2        -0.024630
1    201903              1208.2             1477.2         0.222645
2    201904              1204.6             1439.4         0.194919
```

In [142…
```
final_diff_table_88['YEARMONTH']=final_diff_table_88['YEARMONTH'].astype(int)
stdDev_88=final_diff_table_88[final_diff_table_88['YEARMONTH']>=201902]['percentage
print(f"Standard Deviation of pre-trial percentage diff:{stdDev_88:.4f}")
```

```
Standard Deviation of pre-trial percentage diff:0.1355
```

In [146…
```
from scipy.stats import t
degreesOfFreedom=7 #8-1(trial period months)
trial_data_88= final_diff_table_88.copy()
#Calculate the t-value: t=(x-mu)/stdDev since mu(null hypothesis is 0:
#trial_data['tvalue']=trial_data['percentage_diff']/stdDev
trial_data_88['tValue'] = np.where(stdDev_88!=0,trial_data_88['percentage_diff']/st
#find the 95th percentile of the t-distribution
t_percentile_95=t.ppf(0.95,degreesOfFreedom)
print(f"(95th percentile t-value (df=7): {t_percentile_95:.4f}")
print("\nTrial Month t-values:")
print(trial_data_88[['YEARMONTH','percentage_diff','tValue']])
```

```
(95th percentile t-value (df=7): 1.8946

Trial Month t-values:
   YEARMONTH  percentage_diff    tValue
0     201902        -0.024630 -0.181808
1     201903         0.222645  1.643481
2     201904         0.194919  1.438820
```

In [154…
```
trial_ass88_df= create_assessment_data(metrics_scaled_df,stdDev_88)
print(trial_ass88_df)
```

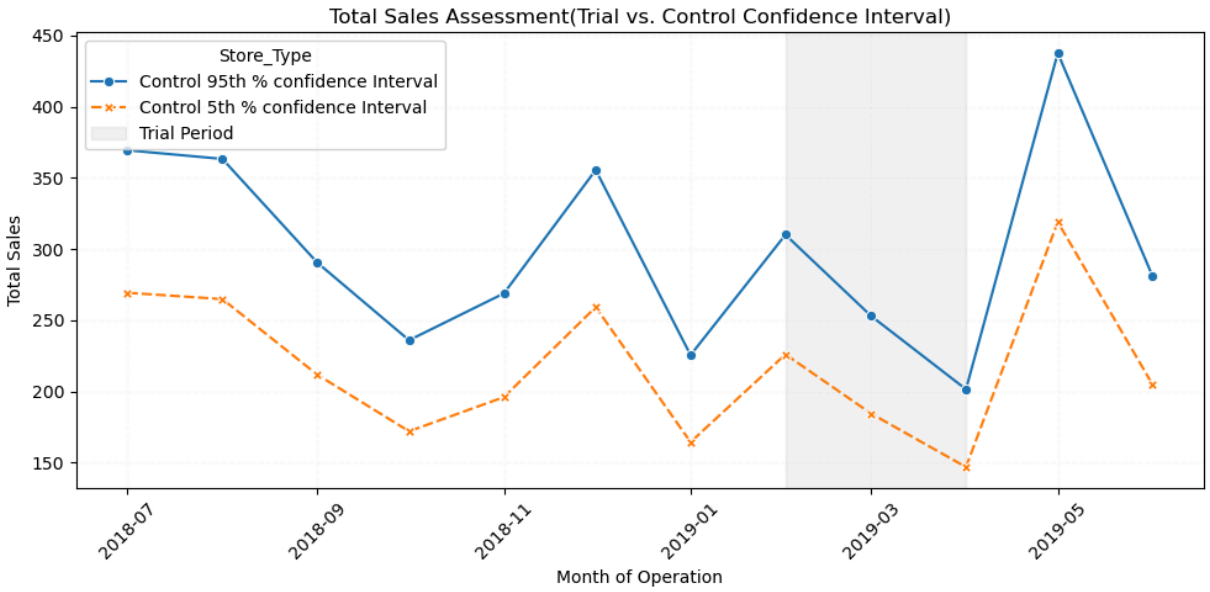
	STORE_NBR	YEARMONTH	monthly_sales	monthly_customers	transactions	\
0	233	201807	369.463284	51	54	
1	233	201808	363.362755	48	50	
2	233	201809	290.537691	42	45	
3	233	201810	236.014214	35	36	
4	233	201811	268.931651	40	41	
5	233	201812	355.610000	47	50	
6	233	201901	225.592477	35	35	
7	233	201902	310.110221	45	47	
8	233	201903	253.044857	40	41	
9	233	201904	201.571644	30	33	
10	233	201905	437.712952	57	62	
11	233	201906	280.878520	41	41	
12	233	201807	269.359598	51	54	
13	233	201808	264.911968	48	50	
14	233	201809	211.818383	42	45	
15	233	201810	172.067689	35	36	
16	233	201811	196.066360	40	41	
17	233	201812	259.259771	47	50	
18	233	201901	164.469655	35	35	
19	233	201902	226.087863	45	47	
20	233	201903	184.483990	40	41	
21	233	201904	146.957111	30	33	
22	233	201905	319.117459	57	62	
23	233	201906	204.776302	41	41	

	chips_per_customer	avg_price_per_unit	scaledControlSales	\
0	1.058824	5.383333	295.311377	
1	1.041667	5.718000	290.435235	
2	1.071429	5.080000	232.226284	
3	1.028571	5.158333	188.645761	
4	1.025000	5.160976	214.956613	
5	1.063830	5.596000	284.238470	
6	1.000000	5.071429	180.315684	
7	1.044444	5.191489	247.870575	
8	1.025000	4.856098	202.258325	
9	1.100000	4.806061	161.115874	
10	1.087719	5.554839	349.863221	
11	1.000000	5.390244	224.505725	
12	1.058824	5.383333	295.311377	
13	1.041667	5.718000	290.435235	
14	1.071429	5.080000	232.226284	
15	1.028571	5.158333	188.645761	
16	1.025000	5.160976	214.956613	
17	1.063830	5.596000	284.238470	
18	1.000000	5.071429	180.315684	
19	1.044444	5.191489	247.870575	
20	1.025000	4.856098	202.258325	
21	1.100000	4.806061	161.115874	
22	1.087719	5.554839	349.863221	
23	1.000000	5.390244	224.505725	

	Store_Type	Transaction_Month
0	Control 95th % confidence Interval	2018-07-01
1	Control 95th % confidence Interval	2018-08-01
2	Control 95th % confidence Interval	2018-09-01

```
3   Control 95th % confidence Interval        2018-10-01
4   Control 95th % confidence Interval        2018-11-01
5   Control 95th % confidence Interval        2018-12-01
6   Control 95th % confidence Interval        2019-01-01
7   Control 95th % confidence Interval        2019-02-01
8   Control 95th % confidence Interval        2019-03-01
9   Control 95th % confidence Interval        2019-04-01
10  Control 95th % confidence Interval        2019-05-01
11  Control 95th % confidence Interval        2019-06-01
12   Control 5th % confidence Interval        2018-07-01
13   Control 5th % confidence Interval        2018-08-01
14   Control 5th % confidence Interval        2018-09-01
15   Control 5th % confidence Interval        2018-10-01
16   Control 5th % confidence Interval        2018-11-01
17   Control 5th % confidence Interval        2018-12-01
18   Control 5th % confidence Interval        2019-01-01
19   Control 5th % confidence Interval        2019-02-01
20   Control 5th % confidence Interval        2019-03-01
21   Control 5th % confidence Interval        2019-04-01
22   Control 5th % confidence Interval        2019-05-01
23   Control 5th % confidence Interval        2019-06-01
```

In [156…
```
plot_assessment(trial_ass88_df)
```



In [158…
```
scaling_factor88_cust=calculate_scaling_factor_cust(pre_trial,88,237)
print(scaling_factor88_cust)
```

```
0.9930761622156281
```

In [160…
```
metrics_scaledCust88_df=apply_scaling_factor_cust(metrics,237,scaling_factor88_cust
print(metrics_scaledCust88_df)
```

```
      STORE_NBR YEARMONTH  monthly_sales  monthly_customers  transactions  \
0             1    201807          206.9                 49            52
1             1    201808          176.1                 42            43
2             1    201809          278.8                 59            62
3             1    201810          188.1                 44            45
4             1    201811          192.6                 46            47
...         ...       ...            ...                ...           ...
3164        272    201902          395.5                 45            48
3165        272    201903          442.3                 50            53
3166        272    201904          445.1                 54            56
3167        272    201905          314.6                 34            40
3168        272    201906          312.1                 34            37

      chips_per_customer  avg_price_per_unit  scaledControlCust
0               1.061224            3.978846               49.0
1               1.023810            4.095349               42.0
2               1.050847            4.496774               59.0
3               1.022727            4.180000               44.0
4               1.021739            4.097872               46.0
...                  ...                 ...                ...
3164            1.066667            8.239583               45.0
3165            1.060000            8.345283               50.0
3166            1.037037            7.948214               54.0
3167            1.176471            7.865000               34.0
3168            1.088235            8.435135               34.0

[3169 rows x 8 columns]
```

In [162…
```
final_diff_table88_cust=calc_percentage_diff_cust(metrics_scaledCust88_df,88,237)
print(final_diff_table88_cust)
```

```
  YEARMONTH  ScaledControlCust  TrialCust_Actual  percentage_diff_cust
0    201902              126.0               124             -0.015873
1    201903              119.0               134              0.126050
2    201904              120.0               128              0.066667
```

In [164…
```
final_diff_table88_cust['YEARMONTH']=final_diff_table88_cust['YEARMONTH'].astype(in
stdDev88_cust=final_diff_table88_cust[final_diff_table88_cust['YEARMONTH']>=201902]
print(f"Standard Deviation of pre-trial percentage diff:{stdDev88_cust:.4f}")
```

```
Standard Deviation of pre-trial percentage diff:0.0713
```

In [166…
```
trial_ass_cust88_df= create_assessment_data_cust(metrics_scaledCust88_df,stdDev88_c
print(trial_ass_cust88_df)
```

	STORE_NBR	YEARMONTH	monthly_sales	monthly_customers	transactions	\
0	233	201807	290.7	58.270138	54	
1	233	201808	285.9	54.842483	50	
2	233	201809	228.6	47.987172	45	
3	233	201810	185.7	39.989310	36	
4	233	201811	211.6	45.702069	41	
5	233	201812	279.8	53.699931	50	
6	233	201901	177.5	39.989310	35	
7	233	201902	244.0	51.414828	47	
8	233	201903	199.1	45.702069	41	
9	233	201904	158.6	34.276552	33	
10	233	201905	344.4	65.125448	62	
11	233	201906	221.0	46.844621	41	
12	233	201807	290.7	49.963629	54	
13	233	201808	285.9	47.024592	50	
14	233	201809	228.6	41.146518	45	
15	233	201810	185.7	34.288765	36	
16	233	201811	211.6	39.187160	41	
17	233	201812	279.8	46.044913	50	
18	233	201901	177.5	34.288765	35	
19	233	201902	244.0	44.085555	47	
20	233	201903	199.1	39.187160	41	
21	233	201904	158.6	29.390370	33	
22	233	201905	344.4	55.841703	62	
23	233	201906	221.0	40.166839	41	

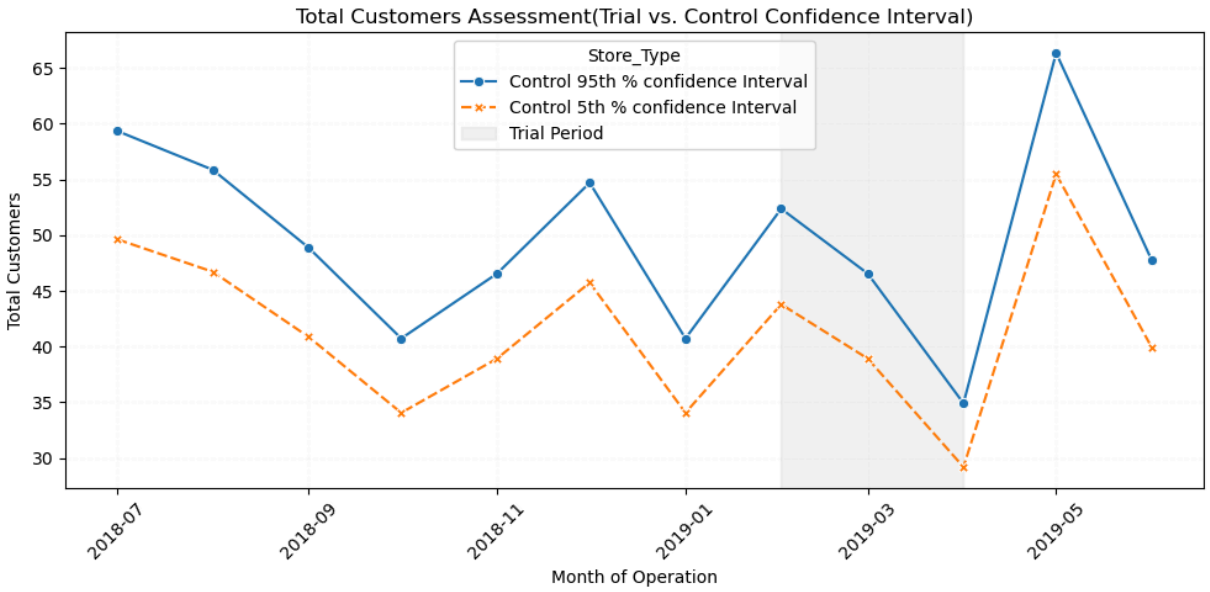
	chips_per_customer	avg_price_per_unit	scaledControlCust	\
0	1.058824	5.383333	50.646884	
1	1.041667	5.718000	47.667656	
2	1.071429	5.080000	41.709199	
3	1.028571	5.158333	34.757666	
4	1.025000	5.160976	39.723046	
5	1.063830	5.596000	46.674580	
6	1.000000	5.071429	34.757666	
7	1.044444	5.191489	44.688427	
8	1.025000	4.856098	39.723046	
9	1.100000	4.806061	29.792285	
10	1.087719	5.554839	56.605341	
11	1.000000	5.390244	40.716123	
12	1.058824	5.383333	50.646884	
13	1.041667	5.718000	47.667656	
14	1.071429	5.080000	41.709199	
15	1.028571	5.158333	34.757666	
16	1.025000	5.160976	39.723046	
17	1.063830	5.596000	46.674580	
18	1.000000	5.071429	34.757666	
19	1.044444	5.191489	44.688427	
20	1.025000	4.856098	39.723046	
21	1.100000	4.806061	29.792285	
22	1.087719	5.554839	56.605341	
23	1.000000	5.390244	40.716123	

	Store_Type	Transaction_Month
0	Control 95th % confidence Interval	2018-07-01
1	Control 95th % confidence Interval	2018-08-01
2	Control 95th % confidence Interval	2018-09-01

```
3   Control 95th % confidence Interval        2018-10-01
4   Control 95th % confidence Interval        2018-11-01
5   Control 95th % confidence Interval        2018-12-01
6   Control 95th % confidence Interval        2019-01-01
7   Control 95th % confidence Interval        2019-02-01
8   Control 95th % confidence Interval        2019-03-01
9   Control 95th % confidence Interval        2019-04-01
10  Control 95th % confidence Interval        2019-05-01
11  Control 95th % confidence Interval        2019-06-01
12   Control 5th % confidence Interval        2018-07-01
13   Control 5th % confidence Interval        2018-08-01
14   Control 5th % confidence Interval        2018-09-01
15   Control 5th % confidence Interval        2018-10-01
16   Control 5th % confidence Interval        2018-11-01
17   Control 5th % confidence Interval        2018-12-01
18   Control 5th % confidence Interval        2019-01-01
19   Control 5th % confidence Interval        2019-02-01
20   Control 5th % confidence Interval        2019-03-01
21   Control 5th % confidence Interval        2019-04-01
22   Control 5th % confidence Interval        2019-05-01
23   Control 5th % confidence Interval        2019-06-01
```

In [168…

```
plot_assessment_cust(trial_ass_cust86_df)
```



In [ ]: