QVI Trial and Control Groups analysis

Here our objective is to find control stores for the selected trial stores. A few stores are selected on which trials are conducted to see if the two metrics notably customer acquisition and revenue are improving or not within the observation period.The control stores need to be selected on the basis of similar metrics for the trial stores and on the basis of the observed similarity score, trial-control combinations will be made.

In [1]:
```
#getting the dataset
import pandas as pd
qvi= pd.read_csv("C:\\Users\\sujoydutta\\Downloads\\QVI_data.csv")
```

In [2]:
```
#seeing the dataset
qvi.head(10)
```

Out[2]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SI
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	1
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	1
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	2
3	1003	2019-03-08	1	4	106	Natural ChipCo Hony Soy Chckn175g	1	3.0	1
4	1004	2018-11-02	1	5	96	WW Original Stacked Chips 160g	1	1.9	1
5	1005	2018-12-28	1	6	86	Cheetos Puffs 165g	1	2.8	1
6	1007	2018-12-04	1	7	49	Infuzions SourCream&Herbs Veg Strws 110g	1	3.8	1
7	1007	2018-12-05	1	8	10	RRD SR Slow Rst Pork Belly 150g	1	2.7	1
8	1009	2018-11-20	1	9	20	Doritos Cheese Supreme 330g	1	5.7	3
9	1010	2018-09-09	1	10	51	Doritos Mexicana 170g	2	8.8	1

In [3]:
```
#examining the dataset
qvi.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 [4]:
```
# Convert DATE to datetime
qvi['DATE'] = pd.to_datetime(qvi['DATE'])
```

In [5]:
```
# Converting PACK_SIZE,PROD_QTY and TOT_SALES to numeric types
qvi['PROD_QTY'] = qvi['PROD_QTY'].astype(int)
qvi['TOT_SALES'] = qvi['TOT_SALES'].astype(float)
qvi['PACK_SIZE'] = qvi['PACK_SIZE'].astype(int)
```

In [6]:
```
# Ensure the remaining columns are of type object
object_columns = ['LYLTY_CARD_NBR', 'LIFESTAGE', 'PREMIUM_CUSTOMER', 'STORE_NBR', 'TXN_I
qvi[object_columns] = qvi[object_columns].astype(object)
```

In [7]:
```
# Display the DataFrame
print(qvi.dtypes)
```

```
LYLTY_CARD_NBR              object
DATE                datetime64[ns]
STORE_NBR                   object
TXN_ID                      object
PROD_NBR                    object
PROD_NAME                   object
PROD_QTY                     int32
TOT_SALES                  float64
PACK_SIZE                    int32
BRAND                       object
LIFESTAGE                   object
PREMIUM_CUSTOMER            object
dtype: object
```

In [8]:
```
# Creating the month ID column
qvi['YEARMONTH'] = qvi['DATE'].dt.year * 100 + qvi['DATE'].dt.month
```

In [9]:
```
# Grouping by STORE_NBR and YEARMONTH and getting the required metrics
measure_over_time = qvi.groupby(['STORE_NBR', 'YEARMONTH']).agg(
    totSales=('TOT_SALES', 'sum'),
    nCustomers=('LYLTY_CARD_NBR', 'nunique'),
    nTxn=('TXN_ID', 'nunique'),
    totQty=('PROD_QTY', 'sum')
).reset_index()
```

In [10]:
```
# Getting additional metrics
measure_over_time['nTxnPerCust'] = measure_over_time['nTxn'] / measure_over_time['nCusto
measure_over_time['nChipsPerTxn'] = measure_over_time['totQty'] / measure_over_time['nTx
measure_over_time['avgPricePerUnit'] = measure_over_time['totSales'] / measure_over_time
```

In [11]:
```
# Sorting by STORE_NBR and YEARMONTH
measure_over_time.sort_values(by=['STORE_NBR', 'YEARMONTH'], inplace=True)
```

In [12]:
```
# Filtering to stores with full observation periods and pre-trial
```

```
stores_with_full_obs = measure_over_time.groupby('STORE_NBR').filter(lambda x: len(x) ==

pre_trial_measures = measure_over_time[(measure_over_time['YEARMONTH'] < 201902) &
                                       (measure_over_time['STORE_NBR'].isin(stores_with_
```

In [13]:
```
#viewing the Pre trial measures which is before February 2019
pre_trial_measures.head()
```

Out[13]:

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTxn	totQty	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	52	62	1.061224	1.192308	3.337097
1	1	201808	176.1	42	43	54	1.023810	1.255814	3.261111
2	1	201809	278.8	59	62	75	1.050847	1.209677	3.717333
3	1	201810	188.1	44	45	58	1.022727	1.288889	3.243103
4	1	201811	192.6	46	47	57	1.021739	1.212766	3.378947

In [14]:
```
#seeing value counts of stores
pre_trial_measures['STORE_NBR'].value_counts()
```

Out[14]:
```
STORE_NBR
1      7
171    7
173    7
174    7
175    7
      ..
98     7
99     7
100    7
101    7
272    7
Name: count, Length: 260, dtype: int64
```

In [15]:
```
#viewing the stores with full observations
stores_with_full_obs
```

Out[15]:
```
array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  12,  13,  14,
        15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,  27,
        28,  29,  30,  32,  33,  34,  35,  36,  37,  38,  39,  40,  41,
        42,  43,  45,  46,  47,  48,  49,  50,  51,  52,  53,  54,  55,
        56,  57,  58,  59,  60,  61,  62,  63,  64,  65,  66,  67,  68,
        69,  70,  71,  72,  73,  74,  75,  77,  78,  79,  80,  81,  82,
        83,  84,  86,  87,  88,  89,  90,  91,  93,  94,  95,  96,  97,
        98,  99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110,
       111, 112, 113, 114, 115, 116, 118, 119, 120, 121, 122, 123, 124,
       125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137,
       138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150,
       151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163,
       164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176,
       177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189,
       190, 191, 192, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203,
       204, 205, 207, 208, 209, 210, 212, 213, 214, 215, 216, 217, 219,
       220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232,
       233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245,
       246, 247, 248, 249, 250, 251, 253, 254, 255, 256, 257, 258, 259,
       260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272],
      dtype=int64)
```

In [16]:
```
#Building a correlation table

def calculate_correlation(input_table, metric_col, store_comparison):
    store_numbers = input_table['STORE_NBR'].unique()
```

```
    correlations = []

    trial_data = input_table[input_table['STORE_NBR'] == store_comparison][metric_col]

    for store in store_numbers:
        if store == store_comparison:
            continue

        control_data = input_table[input_table['STORE_NBR'] == store][metric_col]

        if not trial_data.empty and not control_data.empty:
            corr = trial_data.corr(control_data)
        else:
            corr = None

        correlations.append({'Store1': store_comparison, 'Store2': store, 'corr_measure'

    return pd.DataFrame(correlations)
```

In [17]:
```
#Function to calculate magnitude
def calculate_magnitude_distance(input_table, metric_col, store_comparison):
    calc_dist_df = pd.DataFrame(columns=['Store1', 'Store2', 'YEARMONTH', 'measure'])

    store_numbers = input_table['STORE_NBR'].unique()

    for store in store_numbers:
        if store == store_comparison:
            continue

        trial_data = input_table[input_table['STORE_NBR'] == store_comparison]
        control_data = input_table[input_table['STORE_NBR'] == store]

        merged_data = pd.merge(trial_data[['YEARMONTH', metric_col]],
                               control_data[['YEARMONTH', metric_col]],
                               on='YEARMONTH',
                               suffixes=('_trial', '_control'))

        merged_data['measure'] = abs(merged_data[f'{metric_col}_trial'] - merged_data[f'

        calc_dist_df = pd.concat([
            calc_dist_df,
            pd.DataFrame({
                'Store1': store_comparison,
                'Store2': store,
                'YEARMONTH': merged_data['YEARMONTH'],
                'measure': merged_data['measure']
            })
        ], ignore_index=True)

    min_max_dist = calc_dist_df.groupby(['Store1', 'YEARMONTH']).agg(minDist=('measure',
                                                                              maxDist=('measure',

    dist_df = pd.merge(calc_dist_df, min_max_dist, on=['Store1', 'YEARMONTH'])
    dist_df['magnitudeMeasure'] = 1 - (dist_df['measure'] - dist_df['minDist']) / (dist_

    final_dist_df = dist_df.groupby(['Store1', 'Store2']).agg(mag_measure=('magnitudeMea

    return final_dist_df
```

In [18]:
```
# Calculating for store 77
trial_store1 = 77


corr_nSales77 = calculate_correlation(pre_trial_measures, 'totSales', trial_store1)
corr_nCustomers77 = calculate_correlation(pre_trial_measures, 'nCustomers', trial_store1
```

```
magnitude_nSales77 = calculate_magnitude_distance(pre_trial_measures, 'totSales', trial_
magnitude_nCustomers77 = calculate_magnitude_distance(pre_trial_measures, 'nCustomers',


print("Correlation for total sales:\n", corr_nSales77)
print("\nCorrelation for number of customers:\n", corr_nCustomers77)
print("\nMagnitude distance for total sales:\n", magnitude_nSales77)
print("\nMagnitude distance for number of customers:\n", magnitude_nCustomers77)
```

```
Correlation for total sales:
      Store1  Store2  corr_measure
0         77       1           NaN
1         77       2           NaN
2         77       3           NaN
3         77       4           NaN
4         77       5           NaN
..       ...     ...           ...
254       77     268           NaN
255       77     269           NaN
256       77     270           NaN
257       77     271           NaN
258       77     272           NaN

[259 rows x 3 columns]

Correlation for number of customers:
      Store1  Store2  corr_measure
0         77       1           NaN
1         77       2           NaN
2         77       3           NaN
3         77       4           NaN
4         77       5           NaN
..       ...     ...           ...
254       77     268           NaN
255       77     269           NaN
256       77     270           NaN
257       77     271           NaN
258       77     272           NaN

[259 rows x 3 columns]

Magnitude distance for total sales:
      Store1  Store2  mag_measure
0         77       1     0.955061
1         77       2     0.939318
2         77       3     0.354963
3         77       4     0.177414
4         77       5     0.554066
..       ...     ...          ...
254       77     268     0.962563
255       77     269     0.452903
256       77     270     0.446991
257       77     271     0.553304
258       77     272     0.886697

[259 rows x 3 columns]

Magnitude distance for number of customers:
      Store1  Store2 mag_measure
0         77       1    0.940321
1         77       2    0.924638
2         77       3    0.345067
3         77       4    0.189579
4         77       5    0.481199
```

```
..      ...     ...         ...
254      77     268    0.939907
255      77     269    0.343547
256      77     270    0.357725
257      77     271    0.483457
258      77     272    0.948207

[259 rows x 3 columns]
```

In [19]:
```
# Calculating for store 86
trial_store2 = 86


corr_nSales86 = calculate_correlation(pre_trial_measures, 'totSales', trial_store2)
corr_nCustomers86 = calculate_correlation(pre_trial_measures, 'nCustomers', trial_store2


magnitude_nSales86 = calculate_magnitude_distance(pre_trial_measures, 'totSales', trial_
magnitude_nCustomers86 = calculate_magnitude_distance(pre_trial_measures, 'nCustomers',


print("Correlation for total sales:\n", corr_nSales86)
print("\nCorrelation for number of customers:\n", corr_nCustomers86)
print("\nMagnitude distance for total sales:\n", magnitude_nSales86)
print("\nMagnitude distance for number of customers:\n", magnitude_nCustomers86)
```

```
Correlation for total sales:
      Store1  Store2  corr_measure
0         86       1           NaN
1         86       2           NaN
2         86       3           NaN
3         86       4           NaN
4         86       5           NaN
..       ...     ...           ...
254       86     268           NaN
255       86     269           NaN
256       86     270           NaN
257       86     271           NaN
258       86     272           NaN

[259 rows x 3 columns]

Correlation for number of customers:
      Store1  Store2  corr_measure
0         86       1           NaN
1         86       2           NaN
2         86       3           NaN
3         86       4           NaN
4         86       5           NaN
..       ...     ...           ...
254       86     268           NaN
255       86     269           NaN
256       86     270           NaN
257       86     271           NaN
258       86     272           NaN

[259 rows x 3 columns]

Magnitude distance for total sales:
      Store1  Store2  mag_measure
0         86       1     0.220565
1         86       2     0.179640
2         86       3     0.762894
3         86       4     0.498526
4         86       5     0.929321
..       ...     ...          ...
```

```
254       86     268     0.250819
255       86     269     0.902040
256       86     270     0.834520
257       86     271     0.922919
258       86     272     0.446702

[259 rows x 3 columns]

Magnitude distance for number of customers:
      Store1  Store2 mag_measure
0         86       1    0.444597
1         86       2     0.38062
2         86       3     0.91185
3         86       4    0.773922
4         86       5    0.926509
..       ...     ...         ...
254       86     268     0.42739
255       86     269    0.917082
256       86     270    0.890489
257       86     271    0.935896
258       86     272    0.425196

[259 rows x 3 columns]
```

In [20]:

```
# Calculating for store 88
trial_store3 = 88


corr_nSales88 = calculate_correlation(pre_trial_measures, 'totSales', trial_store3)
corr_nCustomers88 = calculate_correlation(pre_trial_measures, 'nCustomers', trial_store3


magnitude_nSales88 = calculate_magnitude_distance(pre_trial_measures, 'totSales', trial_
magnitude_nCustomers88 = calculate_magnitude_distance(pre_trial_measures, 'nCustomers',


print("Correlation for total sales:\n", corr_nSales88)
print("\nCorrelation for number of customers:\n", corr_nCustomers88)
print("\nMagnitude distance for total sales:\n", magnitude_nSales88)
print("\nMagnitude distance for number of customers:\n", magnitude_nCustomers88)
```

```
Correlation for total sales:
      Store1  Store2  corr_measure
0         88       1           NaN
1         88       2           NaN
2         88       3           NaN
3         88       4           NaN
4         88       5           NaN
..       ...     ...           ...
254       88     268           NaN
255       88     269           NaN
256       88     270           NaN
257       88     271           NaN
258       88     272           NaN

[259 rows x 3 columns]

Correlation for number of customers:
      Store1  Store2  corr_measure
0         88       1           NaN
1         88       2           NaN
2         88       3           NaN
3         88       4           NaN
4         88       5           NaN
..       ...     ...           ...
254       88     268           NaN
```

```
255       88     269           NaN
256       88     270           NaN
257       88     271           NaN
258       88     272           NaN

[259 rows x 3 columns]

Magnitude distance for total sales:
      Store1  Store2  mag_measure
0         88       1     0.143453
1         88       2     0.116355
2         88       3     0.806064
3         88       4     0.901383
4         88       5     0.612614
..       ...     ...          ...
254       88     268     0.161613
255       88     269     0.712728
256       88     270     0.717650
257       88     271     0.615957
258       88     272     0.291095

[259 rows x 3 columns]

Magnitude distance for number of customers:
      Store1  Store2 mag_measure
0         88       1    0.353668
1         88       2    0.302289
2         88       3    0.849307
3         88       4     0.93093
4         88       5    0.742127
..       ...     ...         ...
254       88     268    0.337873
255       88     269    0.852599
256       88     270    0.839071
257       88     271    0.743121
258       88     272    0.336616

[259 rows x 3 columns]
```

In [21]:

```python
# getting the final dataframe
magnitude_df_88 = pd.merge(
    magnitude_nSales88[['Store1', 'Store2', 'mag_measure']],
    magnitude_nCustomers88[['Store1', 'Store2', 'mag_measure']],
    on=['Store1', 'Store2'],
    suffixes=('_sales', '_customers')
)
magnitude_df_88['Trial_Store'] = 88

magnitude_df_86 = pd.merge(
    magnitude_nSales86[['Store1', 'Store2', 'mag_measure']],
    magnitude_nCustomers86[['Store1', 'Store2', 'mag_measure']],
    on=['Store1', 'Store2'],
    suffixes=('_sales', '_customers')
)
magnitude_df_86['Trial_Store'] = 86

magnitude_df_77 = pd.merge(
    magnitude_nSales77[['Store1', 'Store2', 'mag_measure']],
    magnitude_nCustomers77[['Store1', 'Store2', 'mag_measure']],
    on=['Store1', 'Store2'],
    suffixes=('_sales', '_customers')
)
magnitude_df_77['Trial_Store'] = 77
```

```
final_magnitude_df = pd.concat([magnitude_df_88, magnitude_df_86, magnitude_df_77], igno


print("Combined Magnitude Distances DataFrame:\n", final_magnitude_df)
```

```
Combined Magnitude Distances DataFrame:
      Store1  Store2  mag_measure_sales mag_measure_customers  Trial_Store
0         88       1           0.143453               0.353668           88
1         88       2           0.116355               0.302289           88
2         88       3           0.806064               0.849307           88
3         88       4           0.901383                0.93093           88
4         88       5           0.612614               0.742127           88
..       ...     ...                ...                    ...          ...
772       77     268           0.962563               0.939907           77
773       77     269           0.452903               0.343547           77
774       77     270           0.446991               0.357725           77
775       77     271           0.553304               0.483457           77
776       77     272           0.886697               0.948207           77

[777 rows x 5 columns]
```

In [22]:
```
# Defin the correlation weight
corr_weight = 0.5


final_magnitude_df['composite_score'] = (
    final_magnitude_df['mag_measure_sales'] * corr_weight +
    final_magnitude_df['mag_measure_customers'] * (1 - corr_weight)
)
```

In [23]:
```
# Ensuring the composite_score column is numeric and handle NaNs
final_magnitude_df['composite_score'] = pd.to_numeric(final_magnitude_df['composite_scor
```

In [24]:
```
# Grouping by trial store and find the best control store for each trial store
best_control_stores = final_magnitude_df.groupby('Trial_Store').apply(
    lambda x: x.loc[x['composite_score'].idxmax()]
).reset_index(drop=True)
```

In [25]:
```
# Selecting only the relevant columns
best_control_stores = best_control_stores[['Trial_Store', 'Store2', 'composite_score']]
best_control_stores
```

Out[25]:

	Trial_Store	Store2	composite_score
0	77	233	0.989932
1	86	155	0.974909
2	88	237	0.977147

In [26]:
```
# Rename columns for clarity
best_control_stores.columns = ['Trial_Store', 'Control_Store', 'Composite_Score']
```

In [27]:
```
# Printing the best control store for each trial store
print("Best Control Store for Each Trial Store:\n", best_control_stores)
```

```
Best Control Store for Each Trial Store:
    Trial_Store  Control_Store  Composite_Score
0           77            233         0.989932
1           86            155         0.974909
2           88            237         0.977147
```

In [28]:
```
# Getting unique trial and control stores
trial_stores = best_control_stores['Trial_Store'].tolist()
```

```
control_stores = best_control_stores['Control_Store'].tolist()
all_stores = set(trial_stores + control_stores)
all_stores
```

Out[28]:
```
{77, 86, 88, 155, 233, 237}
```

In [29]:
```
# Filtering the main DataFrame
relevant = pre_trial_measures[pre_trial_measures['STORE_NBR'].isin(all_stores)]
```

In [30]:
```
#seeing new database
relevant.head()
```

Out[30]:

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTxn	totQty	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
880	77	201807	296.8	51	55	84	1.078431	1.527273	3.533333
881	77	201808	255.5	47	48	74	1.021277	1.541667	3.452703
882	77	201809	225.2	42	44	70	1.047619	1.590909	3.217143
883	77	201810	204.5	37	38	52	1.027027	1.368421	3.932692
884	77	201811	245.3	41	44	67	1.073171	1.522727	3.661194

Analysing the First Trial and control pair (77 and 233)

We will analyse on the basis of total sales and number of customers.

In [31]:
```
# Defining trial and control stores
trial_store = 77
control_store = 233
```

In [32]:
```
# Adding Store_type based on STORE_NBR
measure_over_time['Store_type'] = measure_over_time['STORE_NBR'].apply(
    lambda x: 'Trial' if x == trial_store else ('Control' if x == control_store else 'Ot
)
```

In [33]:
```
# Calculating average total sales by YEARMONTH and Store_type
pastSales = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'totSales': 'mea
pastSales
```

Out[33]:

	YEARMONTH	Store_type	totSales
0	201807	Control	290.700000
1	201807	Other stores	623.817424
2	201807	Trial	296.800000
3	201808	Control	285.900000
4	201808	Other stores	603.600192
5	201808	Trial	255.500000
6	201809	Control	228.600000
7	201809	Other stores	610.947328
8	201809	Trial	225.200000
9	201810	Control	185.700000
10	201810	Other stores	623.671103
11	201810	Trial	204.500000

12	201811	Control	211.600000
13	201811	Other stores	609.835115
14	201811	Trial	245.300000
15	201812	Control	279.800000
16	201812	Other stores	641.250192
17	201812	Trial	267.300000
18	201901	Control	177.500000
19	201901	Other stores	621.687356
20	201901	Trial	204.400000
21	201902	Control	244.000000
22	201902	Other stores	573.229008
23	201902	Trial	235.000000
24	201903	Control	199.100000
25	201903	Other stores	630.371103
26	201903	Trial	278.500000
27	201904	Control	158.600000
28	201904	Other stores	606.171103
29	201904	Trial	263.500000
30	201905	Control	344.400000
31	201905	Other stores	597.984483
32	201905	Trial	299.300000
33	201906	Control	221.000000
34	201906	Other stores	610.888931
35	201906	Trial	264.700000

In [34]:
```
# Converting YEARMONTH to datetime format
pastSales['TransactionMonth'] = pd.to_datetime(pastSales['YEARMONTH'].astype(str) + '01'
pastSales['TransactionMonth']
```

Out[34]:
```
0     2018-07-01
1     2018-07-01
2     2018-07-01
3     2018-08-01
4     2018-08-01
5     2018-08-01
6     2018-09-01
7     2018-09-01
8     2018-09-01
9     2018-10-01
10    2018-10-01
11    2018-10-01
12    2018-11-01
13    2018-11-01
14    2018-11-01
15    2018-12-01
16    2018-12-01
17    2018-12-01
```

```
18   2019-01-01
19   2019-01-01
20   2019-01-01
21   2019-02-01
22   2019-02-01
23   2019-02-01
24   2019-03-01
25   2019-03-01
26   2019-03-01
27   2019-04-01
28   2019-04-01
29   2019-04-01
30   2019-05-01
31   2019-05-01
32   2019-05-01
33   2019-06-01
34   2019-06-01
35   2019-06-01
Name: TransactionMonth, dtype: datetime64[ns]
```

In [35]:
```
# Calculating average number of customers by YEARMONTH and Store_type
customersacquired = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'nCustom
customersacquired
```

Out[35]:

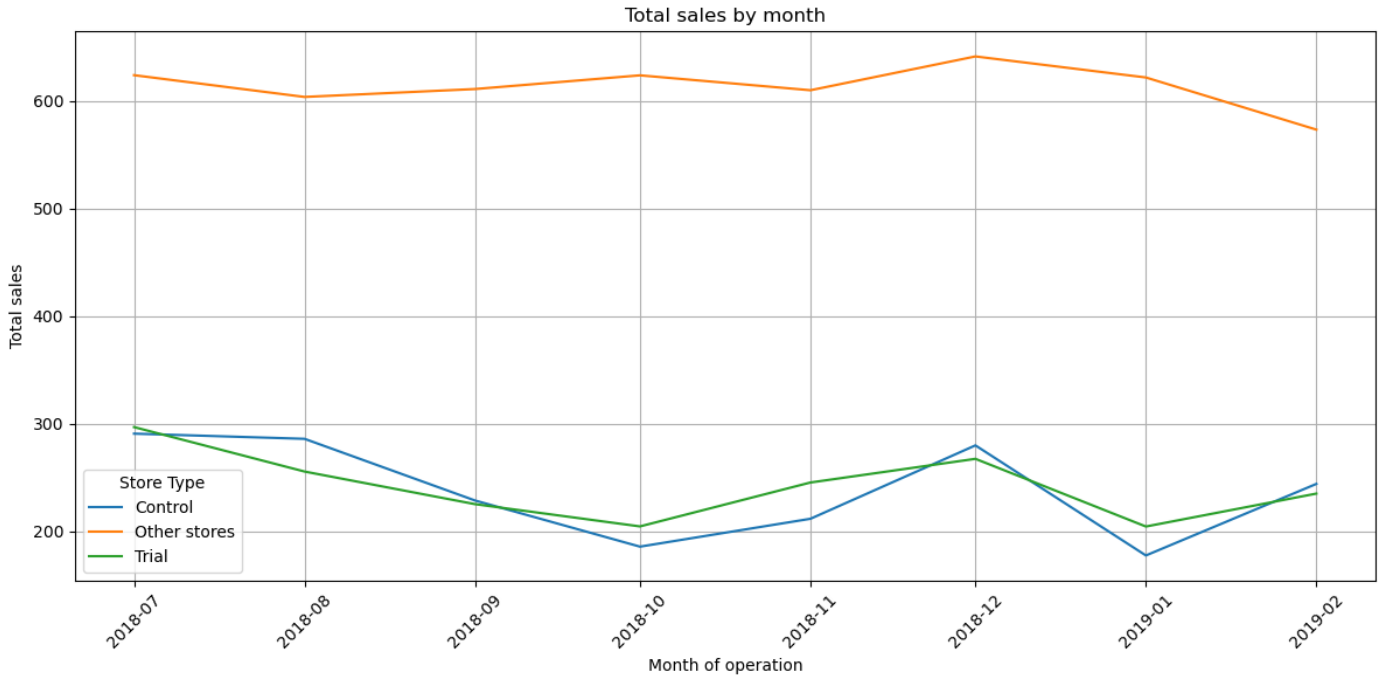
	YEARMONTH	Store_type	nCustomers
0	201807	Control	51.000000
1	201807	Other stores	70.750000
2	201807	Trial	51.000000
3	201808	Control	48.000000
4	201808	Other stores	71.352490
5	201808	Trial	47.000000
6	201809	Control	42.000000
7	201809	Other stores	69.110687
8	201809	Trial	42.000000
9	201810	Control	35.000000
10	201810	Other stores	70.334601
11	201810	Trial	37.000000
12	201811	Control	40.000000
13	201811	Other stores	69.534351
14	201811	Trial	41.000000
15	201812	Control	47.000000
16	201812	Other stores	72.731801
17	201812	Trial	46.000000
18	201901	Control	35.000000
19	201901	Other stores	70.471264
20	201901	Trial	35.000000
21	201902	Control	45.000000
22	201902	Other stores	65.492366

23	201902	Trial	45.000000
24	201903	Control	40.000000
25	201903	Other stores	71.509506
26	201903	Trial	50.000000
27	201904	Control	30.000000
28	201904	Other stores	68.771863
29	201904	Trial	47.000000
30	201905	Control	57.000000
31	201905	Other stores	70.865900
32	201905	Trial	55.000000
33	201906	Control	41.000000
34	201906	Other stores	69.396947
35	201906	Trial	41.000000

In [36]:
```
# Converting YEARMONTH to datetime format
customersacquired['AcquireMonth'] = pd.to_datetime(customersacquired['YEARMONTH'].astype
customersacquired['AcquireMonth']
```

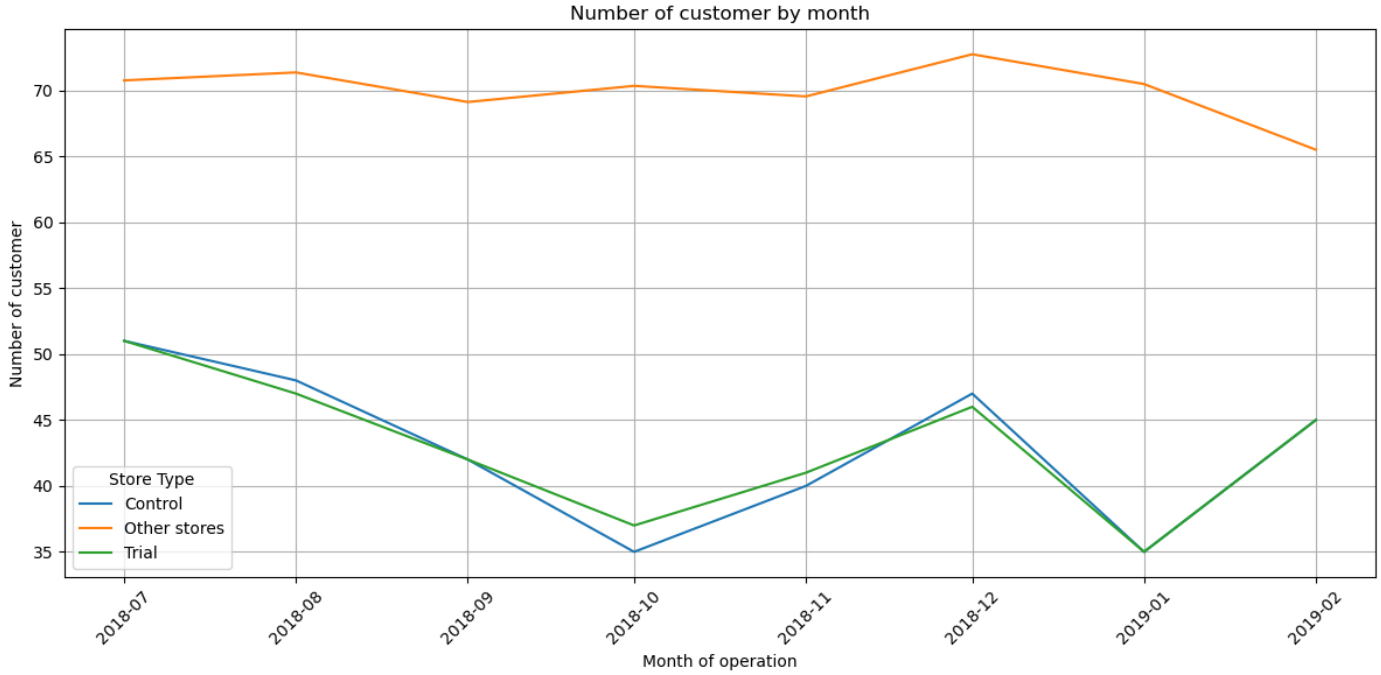
Out[36]:
```
0    2018-07-01
1    2018-07-01
2    2018-07-01
3    2018-08-01
4    2018-08-01
5    2018-08-01
6    2018-09-01
7    2018-09-01
8    2018-09-01
9    2018-10-01
10   2018-10-01
11   2018-10-01
12   2018-11-01
13   2018-11-01
14   2018-11-01
15   2018-12-01
16   2018-12-01
17   2018-12-01
18   2019-01-01
19   2019-01-01
20   2019-01-01
21   2019-02-01
22   2019-02-01
23   2019-02-01
24   2019-03-01
25   2019-03-01
26   2019-03-01
27   2019-04-01
28   2019-04-01
29   2019-04-01
30   2019-05-01
31   2019-05-01
32   2019-05-01
33   2019-06-01
34   2019-06-01
35   2019-06-01
Name: AcquireMonth, dtype: datetime64[ns]
```

In [37]:
```
# Plot the data on basis of total sales by month
import matplotlib.pyplot as plt
import seaborn as sns
pastSales = pastSales[pastSales['YEARMONTH'] < 201903]


plt.figure(figsize=(12, 6))
sns.lineplot(data=pastSales, x='TransactionMonth', y='totSales', hue='Store_type')
plt.xlabel('Month of operation')
plt.ylabel('Total sales')
plt.title('Total sales by month')
plt.legend(title='Store Type')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Before the Trial period we can see the control and trial stores have almost similar sales where control is performing better mostly than trial.

In [38]:
```
# Plotting the data on basis of number of customers by month

customersacquired = customersacquired[customersacquired['YEARMONTH'] < 201903]


plt.figure(figsize=(12, 6))
sns.lineplot(data=customersacquired, x='AcquireMonth', y='nCustomers', hue='Store_type')
plt.xlabel('Month of operation')
plt.ylabel('Number of customer')
plt.title('Number of customer by month')
plt.legend(title='Store Type')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Here we can see that the number of customers acquired is almost the same for both types of stores.

In [39]:
```
# Sampling pre-trial measures for demonstration
pre_trial_period = measure_over_time[(measure_over_time['STORE_NBR'] == trial_store) |
                                     (measure_over_time['STORE_NBR'] == control_store
pre_trial_period
```

Out[39]:

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTxn	totQty	nTxnPerCust	nChipsPerTxn	avgPricePerUn
880	77	201807	296.8	51	55	84	1.078431	1.527273	3.53333
881	77	201808	255.5	47	48	74	1.021277	1.541667	3.45270
882	77	201809	225.2	42	44	70	1.047619	1.590909	3.21714
883	77	201810	204.5	37	38	52	1.027027	1.368421	3.93269
884	77	201811	245.3	41	44	67	1.073171	1.522727	3.66119
885	77	201812	267.3	46	48	72	1.043478	1.500000	3.71250
886	77	201901	204.4	35	39	65	1.114286	1.666667	3.14461
887	77	201902	235.0	45	45	74	1.000000	1.644444	3.17567
888	77	201903	278.5	50	55	82	1.100000	1.490909	3.39634
889	77	201904	263.5	47	48	78	1.021277	1.625000	3.37820
890	77	201905	299.3	55	56	84	1.018182	1.500000	3.56309
891	77	201906	264.7	41	42	70	1.024390	1.666667	3.78142
2699	233	201807	290.7	51	54	88	1.058824	1.629630	3.30340
2700	233	201808	285.9	48	50	80	1.041667	1.600000	3.57375
2701	233	201809	228.6	42	45	70	1.071429	1.555556	3.26571
2702	233	201810	185.7	35	36	56	1.028571	1.555556	3.31607
2703	233	201811	211.6	40	41	62	1.025000	1.512195	3.41290
2704	233	201812	279.8	47	50	75	1.063830	1.500000	3.73066
2705	233	201901	177.5	35	35	47	1.000000	1.342857	3.77659

2706	233	201902	244.0	45	47	70	1.044444	1.489362	3.48571
2707	233	201903	199.1	40	41	59	1.025000	1.439024	3.37457
2708	233	201904	158.6	30	32	46	1.066667	1.437500	3.44782
2709	233	201905	344.4	57	62	92	1.087719	1.483871	3.74347
2710	233	201906	221.0	41	41	61	1.000000	1.487805	3.62295

In [92]:
```
# Calculate scaling factor for sales
scalingFactorForControlSales = pre_trial_period[pre_trial_period['Store_type'] == 'Trial
scalingFactorForControlSales
```

Out[92]:
```
1.0753829282960135
```

In [93]:
```
# Apply scaling factor to control store sales
measure_over_time['scaledControlSales'] = measure_over_time.apply(
    lambda row: row['totSales'] * scalingFactorForControlSales if row['STORE_NBR'] == co
    axis=1
)
```

In [42]:
```
# Recalculating pastSales with scaled control sales
pastSales = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'totSales': 'mea
pastSales['TransactionMonth'] = pd.to_datetime(pastSales['YEARMONTH'].astype(str) + '01'
pastSales
```

Out[42]:

	YEARMONTH	Store_type	totSales	scaledControlSales	TransactionMonth
0	201807	Control	290.700000	312.613817	2018-07-01
1	201807	Other stores	623.817424	623.817424	2018-07-01
2	201807	Trial	296.800000	296.800000	2018-07-01
3	201808	Control	285.900000	307.451979	2018-08-01
4	201808	Other stores	603.600192	603.600192	2018-08-01
5	201808	Trial	255.500000	255.500000	2018-08-01
6	201809	Control	228.600000	245.832537	2018-09-01
7	201809	Other stores	610.947328	610.947328	2018-09-01
8	201809	Trial	225.200000	225.200000	2018-09-01
9	201810	Control	185.700000	199.698610	2018-10-01
10	201810	Other stores	623.671103	623.671103	2018-10-01
11	201810	Trial	204.500000	204.500000	2018-10-01
12	201811	Control	211.600000	227.551028	2018-11-01
13	201811	Other stores	609.835115	609.835115	2018-11-01
14	201811	Trial	245.300000	245.300000	2018-11-01
15	201812	Control	279.800000	300.892143	2018-12-01
16	201812	Other stores	641.250192	641.250192	2018-12-01
17	201812	Trial	267.300000	267.300000	2018-12-01
18	201901	Control	177.500000	190.880470	2019-01-01
19	201901	Other stores	621.687356	621.687356	2019-01-01
20	201901	Trial	204.400000	204.400000	2019-01-01

21	201902	Control	244.000000	262.393435	2019-02-01
22	201902	Other stores	573.229008	573.229008	2019-02-01
23	201902	Trial	235.000000	235.000000	2019-02-01
24	201903	Control	199.100000	214.108741	2019-03-01
25	201903	Other stores	630.371103	630.371103	2019-03-01
26	201903	Trial	278.500000	278.500000	2019-03-01
27	201904	Control	158.600000	170.555732	2019-04-01
28	201904	Other stores	606.171103	606.171103	2019-04-01
29	201904	Trial	263.500000	263.500000	2019-04-01
30	201905	Control	344.400000	370.361881	2019-05-01
31	201905	Other stores	597.984483	597.984483	2019-05-01
32	201905	Trial	299.300000	299.300000	2019-05-01
33	201906	Control	221.000000	237.659627	2019-06-01
34	201906	Other stores	610.888931	610.888931	2019-06-01
35	201906	Trial	264.700000	264.700000	2019-06-01

In [72]:
```
# Control store 95th percentile and 5th percentile
stdDev = pastSales['scaledControlSales'].std()
pastSales_Controls95 = pastSales[pastSales['Store_type'] == 'Control'].copy()
pastSales_Controls95['scaledControlSales'] = pastSales_Controls95['scaledControlSales']
pastSales_Controls95['Store_type'] = 'Control 95th % confidence interval'

pastSales_Controls5 = pastSales[pastSales['Store_type'] == 'Control'].copy()
pastSales_Controls5['scaledControlSales'] = pastSales_Controls5['scaledControlSales'] *
pastSales_Controls5['Store_type'] = 'Control 5th % confidence interval'
```

In [44]:
```
# Rename columns before merging to avoid conflicts
pastSales_Controls95.rename(columns={'scaledControlSales': 'scaled95'}, inplace=True)
pastSales_Controls5.rename(columns={'scaledControlSales': 'scaled5'}, inplace=True)
```

In [45]:
```
#Building final df for sales assessment

combined_data_95 = pd.merge(
    pastSales,
    pastSales_Controls95[['TransactionMonth', 'scaled95']],
    on='TransactionMonth'
)


combined_data = pd.merge(
    combined_data_95,
    pastSales_Controls5[['TransactionMonth', 'scaled5']],
    on='TransactionMonth'
)


combined_data.rename(columns={'scaledControlSales': 'scaledNormal'}, inplace=True)


combined_data.drop_duplicates(inplace=True)
```

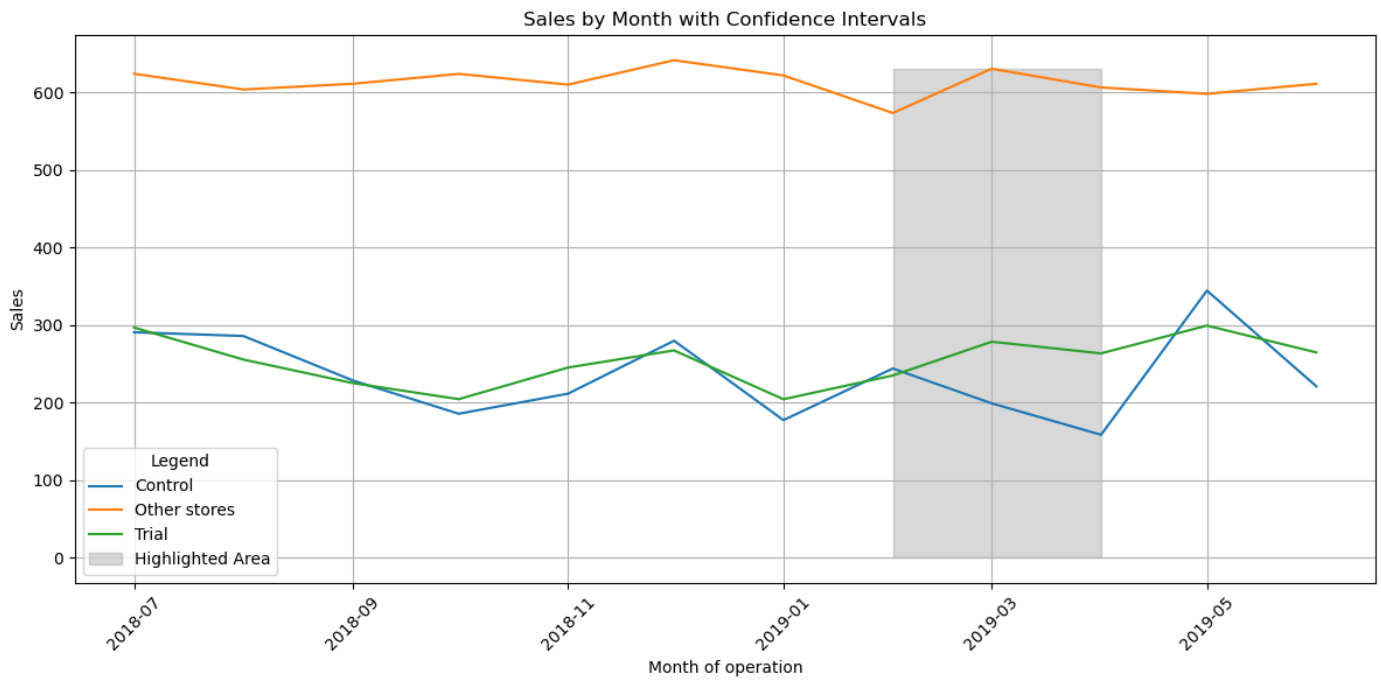
```
combined_data.head()
```

Out[45]:

	YEARMONTH	Store_type	totSales	scaledNormal	TransactionMonth	scaled95	scaled5
0	201807	Control	290.700000	312.613817	2018-07-01	110480.631600	-109855.403965
1	201807	Other stores	623.817424	623.817424	2018-07-01	110480.631600	-109855.403965
2	201807	Trial	296.800000	296.800000	2018-07-01	110480.631600	-109855.403965
3	201808	Control	285.900000	307.451979	2018-08-01	108656.390005	-108041.486046
4	201808	Other stores	603.600192	603.600192	2018-08-01	108656.390005	-108041.486046

In [46]:

```
# Plotting the difference between control and trial stores on sales basis in the trial p

plt.figure(figsize=(12, 6))

sns.lineplot(data=combined_data, x='TransactionMonth', y='totSales', hue='Store_type', p


highlight_period = (combined_data['TransactionMonth'] < pd.to_datetime('2019-05-01')) &
highlight_data = combined_data[highlight_period]

plt.fill_between(
    highlight_data['TransactionMonth'],
    0,
    highlight_data['totSales'].max(),
    color='grey',
    alpha=0.3,
    label='Highlighted Area',
    zorder=1
)


plt.xlabel('Month of operation')
plt.ylabel('Sales')
plt.title('Sales by Month with Confidence Intervals')


handles, labels = plt.gca().get_legend_handles_labels()
unique_labels = []
unique_handles = []
for handle, label in zip(handles, labels):
    if label not in unique_labels:
        unique_labels.append(label)
        unique_handles.append(handle)
plt.legend(handles=unique_handles, labels=unique_labels, title='Legend')

plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()


plt.show()
```



The area shaded in grey represents the highlighted period which is the trial period. The Trial store has seen a significant increase in sales than the control stores within the trial period.

In [47]:
```
# Set trial and control stores
trial_store = 77
control_store = 235
```

In [49]:
```
# Extracting pre-trial measures
preTrialMeasures = measure_over_time[measure_over_time['YEARMONTH'] < 201902]
preTrialMeasures.head()
```

Out[49]:

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTxn	totQty	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	52	62	1.061224	1.192308	3.337097
1	1	201808	176.1	42	43	54	1.023810	1.255814	3.261111
2	1	201809	278.8	59	62	75	1.050847	1.209677	3.717333
3	1	201810	188.1	44	45	58	1.022727	1.288889	3.243103
4	1	201811	192.6	46	47	57	1.021739	1.212766	3.378947

In [102…
```
# Calculate scaling factor for customers
scalingFactorForControlcustomers = pre_trial_period[pre_trial_period['Store_type'] == 'T
scalingFactorForControlcustomers
```

Out[102]:
```
1.050880626223092
```

In [95]:
```
# Applying scaling factor to control store sales
measure_over_time['scaledControlcustomers'] = measure_over_time.apply(
    lambda row: row['nCustomers'] * scalingFactorForControlSales if row['STORE_NBR'] ==
    axis=1
)
```

```
# Recalculating pastcustomer with scaled customer sales
```

In [54]:
```
pastcustomer = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'nCustomers':
pastcustomer['TransactionMonth'] = pd.to_datetime(pastcustomer['YEARMONTH'].astype(str)
pastcustomer.head()
```

Out[54]:

	YEARMONTH	Store_type	nCustomers	scaledControlcustomers	TransactionMonth
0	201807	Control	51.00000	51.000000	2018-07-01
1	201807	Other stores	70.75000	70.765990	2018-07-01
2	201807	Trial	51.00000	51.000000	2018-07-01
3	201808	Control	48.00000	48.000000	2018-08-01
4	201808	Other stores	71.35249	71.365776	2018-08-01

In [65]:
```
# Converting YEARMONTH to TransactionMonth
pastcustomer['TransactionMonth'] = pd.to_datetime(pastcustomer['YEARMONTH'].astype(str)
pastcustomer.head()
```

Out[65]:

	YEARMONTH	Store_type	nCustomers	scaledControlcustomers	TransactionMonth
0	201807	Control	51.00000	51.000000	2018-07-01
1	201807	Other stores	70.75000	70.765990	2018-07-01
2	201807	Trial	51.00000	51.000000	2018-07-01
3	201808	Control	48.00000	48.000000	2018-08-01
4	201808	Other stores	71.35249	71.365776	2018-08-01

In [58]:
```
# Defining highlight period
highlight_start = pd.to_datetime('2019-01-01')
highlight_end = pd.to_datetime('2019-05-01')
highlight_period = (pastcustomer['TransactionMonth'] >= highlight_start) & (pastcustomer
```

In [103…
```
# Control store 95th percentile and 5th percentile
stdDev_nc = pastcustomer[pastcustomer['Store_type'] == 'Control']['nCustomers'].std()

pastcustomer95 = pastcustomer[pastcustomer['Store_type'] == 'Control'].copy()
pastcustomer95['scaledControlcustomers'] = pastcustomer95['scaledControlcustomers'] * (1
pastcustomer95['Store_type'] = 'Control 95th % confidence interval'


pastcustomer5 = pastcustomer[pastcustomer['Store_type'] == 'Control'].copy()
pastcustomer5['scaledControlcustomers'] = pastcustomer5['scaledControlcustomers'] * (1 -
pastcustomer5['Store_type'] = 'Control 5th % confidence interval'
```

In [104…
```
# Rename columns before merging to avoid conflicts
pastcustomer95.rename(columns={'scaledControlcustomers': 'scaledcust95'}, inplace=True)
pastcustomer5.rename(columns={'scaledControlcustomers': 'scaledcust5'}, inplace=True)
```

In [108…
```
#Building final df for customer assessment

pastcustomer95 = pd.merge(
    pastcustomer,
    pastcustomer95[['TransactionMonth', 'scaledcust95']],
    on='TransactionMonth'
)


mergedata = pd.merge(
    pastcustomer95,
```

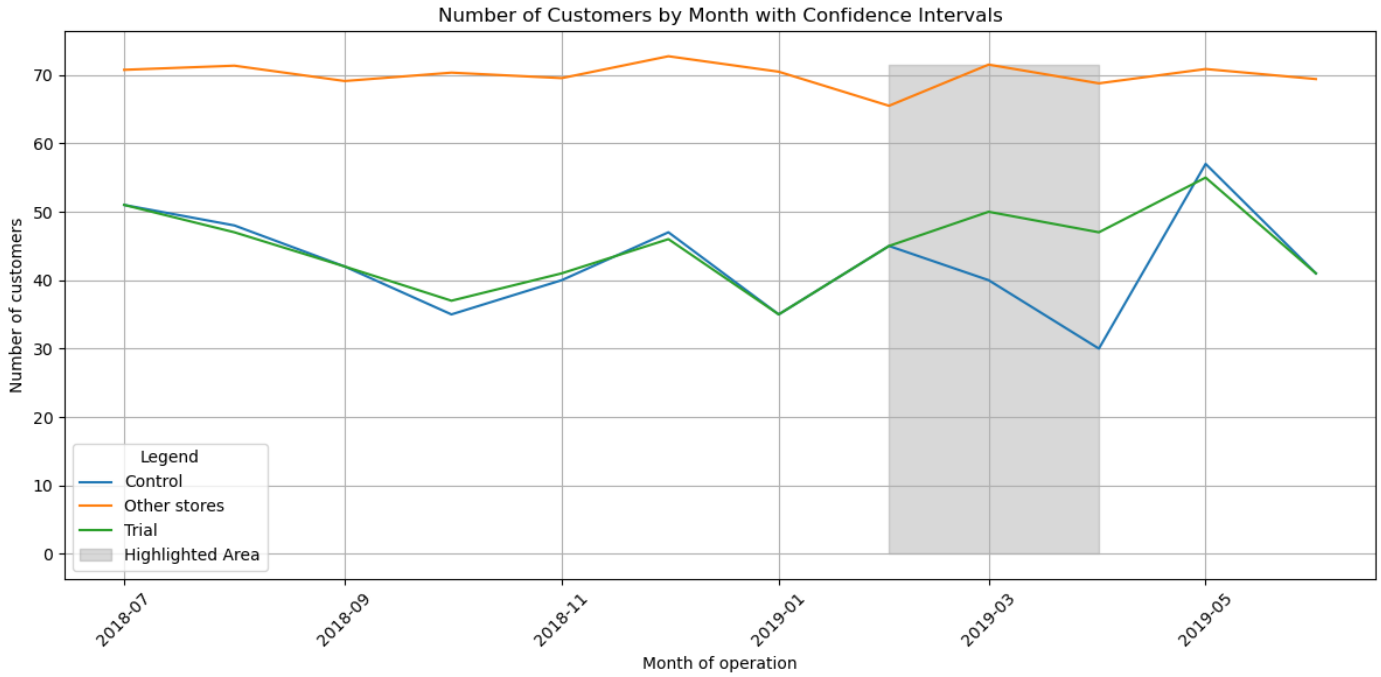
```
    pastcustomer5[['TransactionMonth', 'scaledcust5']],
    on='TransactionMonth'
)


mergedata.rename(columns={'scaledControlcustomers': 'scaledNormal'}, inplace=True)


mergedata.drop_duplicates(inplace=True)


mergedata.head()
```

Out[108]:

	YEARMONTH	Store_type	nCustomers	scaledNormal	TransactionMonth	scaledcust95	scaledcust5
0	201807	Control	51.00000	51.000000	2018-07-01	818.571436	-716.571436
1	201807	Other stores	70.75000	70.765990	2018-07-01	818.571436	-716.571436
2	201807	Trial	51.00000	51.000000	2018-07-01	818.571436	-716.571436
3	201808	Control	48.00000	48.000000	2018-08-01	770.420175	-674.420175
4	201808	Other stores	71.35249	71.365776	2018-08-01	770.420175	-674.420175

In [118…

```
# Plotting the difference between control and trial stores on sales basis in the trial p
plt.figure(figsize=(12, 6))
sns.lineplot(data=mergedata, x='TransactionMonth', y='nCustomers', hue='Store_type', pal

highlight_period = (mergedata['TransactionMonth'] < pd.to_datetime('2019-05-01')) & (mer
highlight_data = mergedata[highlight_period]

plt.fill_between(
    highlight_data['TransactionMonth'],
    0,
    highlight_data['nCustomers'].max(),
    color='grey',
    alpha=0.3,
    label='Highlighted Area',
    zorder=1
)

plt.xlabel('Month of operation')
plt.ylabel('Number of customers')
plt.title('Number of Customers by Month with Confidence Intervals')

handles, labels = plt.gca().get_legend_handles_labels()
unique_labels = []
unique_handles = []
for handle, label in zip(handles, labels):
    if label not in unique_labels:
        unique_labels.append(label)
        unique_handles.append(handle)
plt.legend(handles=unique_handles, labels=unique_labels, title='Legend')

plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```



We can clearly see in the highlight period that the trial store is performing better than the control store within the trial period. The trial stores are getting more customers than the control stores.

Analysing the Second Trial and control pair (86 and 155)

We will analyse on the basis of total sales and number of customers.

In [123…
```
# Set trial and control stores
trial_store = 86
control_store = 155
```

In [124…
```
# Adding Store_type based on STORE_NBR
measure_over_time['Store_type'] = measure_over_time['STORE_NBR'].apply(
    lambda x: 'Trial' if x == trial_store else ('Control' if x == control_store else 'Ot
)
```

Out[124]:

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTxn	totQty	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	52	62	1.061224	1.192308	3.337097
1	1	201808	176.1	42	43	54	1.023810	1.255814	3.261111
2	1	201809	278.8	59	62	75	1.050847	1.209677	3.717333
3	1	201810	188.1	44	45	58	1.022727	1.288889	3.243103
4	1	201811	192.6	46	47	57	1.021739	1.212766	3.378947

In [125…
```
# Calculating average total sales by YEARMONTH and Store_type
pastSales = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'totSales': 'mea
pastSales
```

Out[125]:

	YEARMONTH	Store_type	totSales
0	201807	Control	924.600000

1	201807	Other stores	619.160985
2	201807	Trial	892.200000
3	201808	Control	782.700000
4	201808	Other stores	599.748276
5	201808	Trial	764.050000
6	201809	Control	1014.400000
7	201809	Other stores	605.316794
8	201809	Trial	914.600000
9	201810	Control	963.800000
10	201810	Other stores	617.884030
11	201810	Trial	948.400000
12	201811	Control	898.800000
13	201811	Other stores	604.644656
14	201811	Trial	918.000000
15	201812	Control	849.800000
16	201812	Other stores	636.867433
17	201812	Trial	841.200000
18	201901	Control	874.600000
19	201901	Other stores	616.575862
20	201901	Trial	841.400000
21	201902	Control	891.200000
22	201902	Other stores	568.170229
23	201902	Trial	913.200000
24	201903	Control	804.400000
25	201903	Other stores	625.224335
26	201903	Trial	1026.800000
27	201904	Control	844.600000
28	201904	Other stores	601.339544
29	201904	Trial	848.200000
30	201905	Control	922.850000
31	201905	Other stores	593.507663
32	201905	Trial	889.300000
33	201906	Control	857.200000
34	201906	Other stores	606.272519
35	201906	Trial	838.000000

In [126…

```
# Converting YEARMONTH to datetime format
pastSales['TransactionMonth'] = pd.to_datetime(pastSales['YEARMONTH'].astype(str) + '01'
pastSales['TransactionMonth']
```

Out[126]:
```
0    2018-07-01
1    2018-07-01
2    2018-07-01
3    2018-08-01
4    2018-08-01
5    2018-08-01
6    2018-09-01
7    2018-09-01
8    2018-09-01
9    2018-10-01
10   2018-10-01
11   2018-10-01
12   2018-11-01
13   2018-11-01
14   2018-11-01
15   2018-12-01
16   2018-12-01
17   2018-12-01
18   2019-01-01
19   2019-01-01
20   2019-01-01
21   2019-02-01
22   2019-02-01
23   2019-02-01
24   2019-03-01
25   2019-03-01
26   2019-03-01
27   2019-04-01
28   2019-04-01
29   2019-04-01
30   2019-05-01
31   2019-05-01
32   2019-05-01
33   2019-06-01
34   2019-06-01
35   2019-06-01
Name: TransactionMonth, dtype: datetime64[ns]
```

In [127…
```
# Calculating average number of customers by YEARMONTH and Store_type
customersacquired = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'nCustom
customersacquired
```

Out[127]:

	YEARMONTH	Store_type	nCustomers
0	201807	Control	101.000000
1	201807	Other stores	70.378788
2	201807	Trial	99.000000
3	201808	Control	91.000000
4	201808	Other stores	71.007663
5	201808	Trial	94.000000
6	201809	Control	103.000000
7	201809	Other stores	68.645038
8	201809	Trial	103.000000
9	201810	Control	108.000000
10	201810	Other stores	69.783270
11	201810	Trial	109.000000

12	201811	Control	101.000000
13	201811	Other stores	69.076336
14	201811	Trial	100.000000
15	201812	Control	97.000000
16	201812	Other stores	72.340996
17	201812	Trial	98.000000
18	201901	Control	96.000000
19	201901	Other stores	70.011494
20	201901	Trial	94.000000
21	201902	Control	95.000000
22	201902	Other stores	65.064885
23	201902	Trial	107.000000
24	201903	Control	94.000000
25	201903	Other stores	71.057034
26	201903	Trial	115.000000
27	201904	Control	99.000000
28	201904	Other stores	68.288973
29	201904	Trial	105.000000
30	201905	Control	106.000000
31	201905	Other stores	70.490421
32	201905	Trial	104.000000
33	201906	Control	95.000000
34	201906	Other stores	68.973282
35	201906	Trial	98.000000

In [128…

```
# Converting YEARMONTH to datetime format
customersacquired['AcquireMonth'] = pd.to_datetime(customersacquired['YEARMONTH'].astype
customersacquired['AcquireMonth']
```

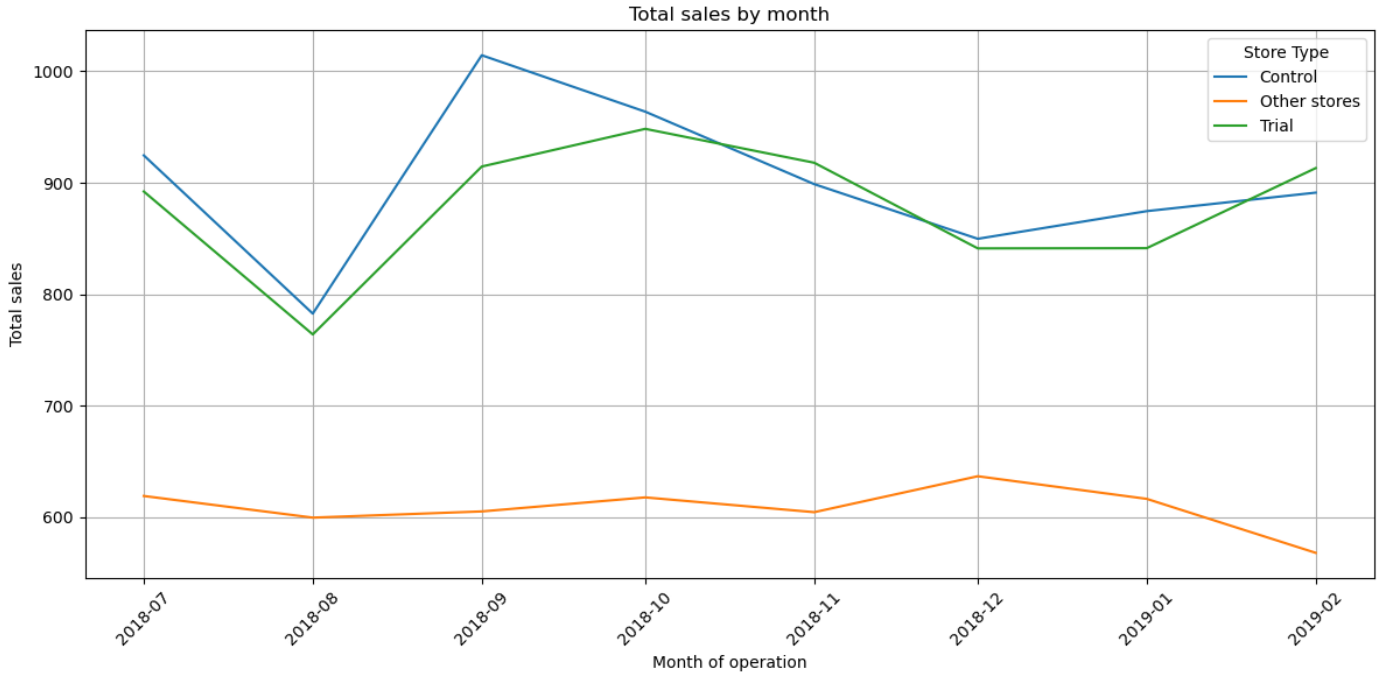
Out[128]:

```
0    2018-07-01
1    2018-07-01
2    2018-07-01
3    2018-08-01
4    2018-08-01
5    2018-08-01
6    2018-09-01
7    2018-09-01
8    2018-09-01
9    2018-10-01
10   2018-10-01
11   2018-10-01
12   2018-11-01
13   2018-11-01
14   2018-11-01
15   2018-12-01
16   2018-12-01
17   2018-12-01
```

```
18   2019-01-01
19   2019-01-01
20   2019-01-01
21   2019-02-01
22   2019-02-01
23   2019-02-01
24   2019-03-01
25   2019-03-01
26   2019-03-01
27   2019-04-01
28   2019-04-01
29   2019-04-01
30   2019-05-01
31   2019-05-01
32   2019-05-01
33   2019-06-01
34   2019-06-01
35   2019-06-01
Name: AcquireMonth, dtype: datetime64[ns]
```

In [130…

```
# Plot the data on basis of total sales by month

pastSales = pastSales[pastSales['YEARMONTH'] < 201903]
plt.figure(figsize=(12, 6))
sns.lineplot(data=pastSales, x='TransactionMonth', y='totSales', hue='Store_type')
plt.xlabel('Month of operation')
plt.ylabel('Total sales')
plt.title('Total sales by month')
plt.legend(title='Store Type')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

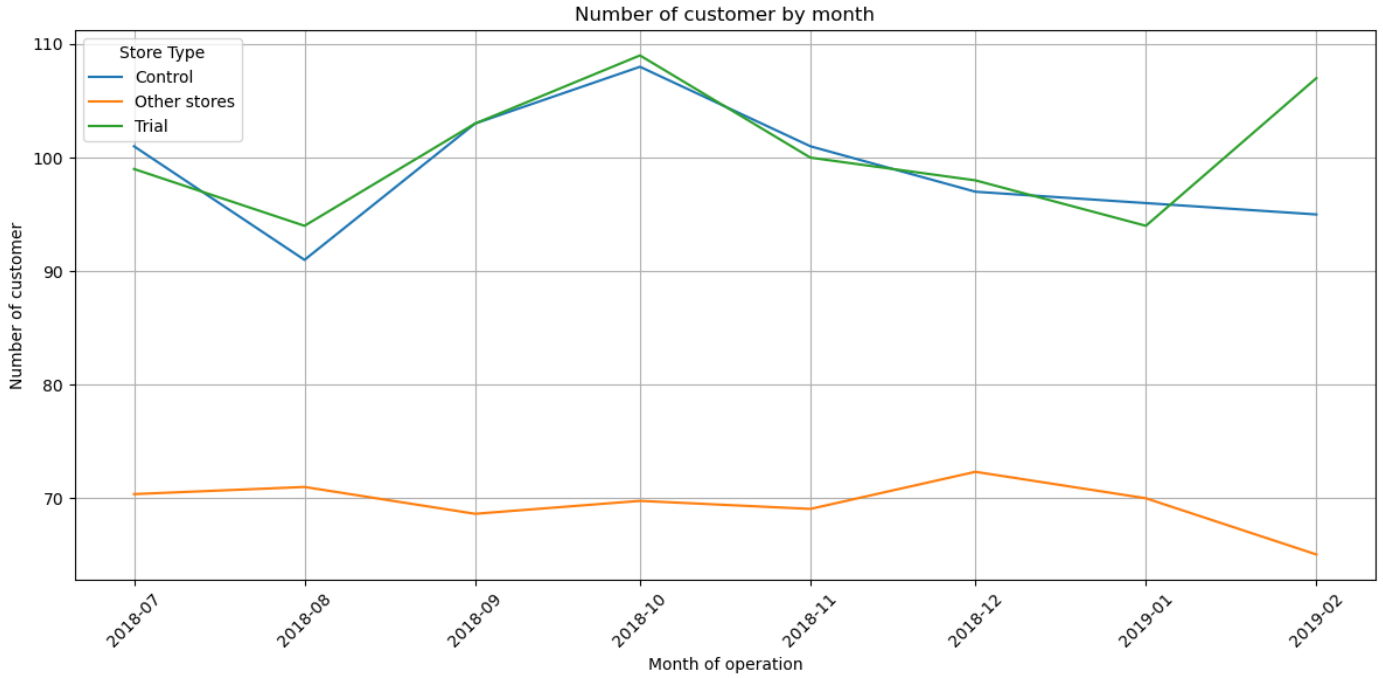


Here we can see the control stores have higher sales than trial stores before the trial period.

In [131…

```
# Plotting the data on basis of number of customers by month

customersacquired = customersacquired[customersacquired['YEARMONTH'] < 201903]


plt.figure(figsize=(12, 6))
sns.lineplot(data=customersacquired, x='AcquireMonth', y='nCustomers', hue='Store_type')
```

```
plt.xlabel('Month of operation')
plt.ylabel('Number of customer')
plt.title('Number of customer by month')
plt.legend(title='Store Type')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Acquisition of customers is almost same in both trial and control stores.

In [132…
```
# Sampling pre-trial measures for demonstration
pre_trial_period = measure_over_time[(measure_over_time['STORE_NBR'] == trial_store) |
                                     (measure_over_time['STORE_NBR'] == control_store
```

In [133…
```
# Calculate scaling factor for sales
scalingFactorForControlSales = pre_trial_period[pre_trial_period['Store_type'] == 'Trial
scalingFactorForControlSales
```

Out[133]:
```
1.000602129090832
```

In [134…
```
# Apply scaling factor to control store sales
measure_over_time['scaledControlSales'] = measure_over_time.apply(
    lambda row: row['totSales'] * scalingFactorForControlSales if row['STORE_NBR'] == co
    axis=1
)
```

In [135…
```
# Recalculating pastSales with scaled control sales
pastSales = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'totSales': 'mea
pastSales['TransactionMonth'] = pd.to_datetime(pastSales['YEARMONTH'].astype(str) + '01'
pastSales
```

Out[135]:

	YEARMONTH	Store_type	totSales	scaledControlSales	TransactionMonth
0	201807	Control	924.600000	925.156729	2018-07-01
1	201807	Other stores	619.160985	619.160985	2018-07-01
2	201807	Trial	892.200000	892.200000	2018-07-01
3	201808	Control	782.700000	783.171286	2018-08-01

4	201808	Other stores	599.748276	599.748276	2018-08-01
5	201808	Trial	764.050000	764.050000	2018-08-01
6	201809	Control	1014.400000	1015.010800	2018-09-01
7	201809	Other stores	605.316794	605.316794	2018-09-01
8	201809	Trial	914.600000	914.600000	2018-09-01
9	201810	Control	963.800000	964.380332	2018-10-01
10	201810	Other stores	617.884030	617.884030	2018-10-01
11	201810	Trial	948.400000	948.400000	2018-10-01
12	201811	Control	898.800000	899.341194	2018-11-01
13	201811	Other stores	604.644656	604.644656	2018-11-01
14	201811	Trial	918.000000	918.000000	2018-11-01
15	201812	Control	849.800000	850.311689	2018-12-01
16	201812	Other stores	636.867433	636.867433	2018-12-01
17	201812	Trial	841.200000	841.200000	2018-12-01
18	201901	Control	874.600000	875.126622	2019-01-01
19	201901	Other stores	616.575862	616.575862	2019-01-01
20	201901	Trial	841.400000	841.400000	2019-01-01
21	201902	Control	891.200000	891.736617	2019-02-01
22	201902	Other stores	568.170229	568.170229	2019-02-01
23	201902	Trial	913.200000	913.200000	2019-02-01
24	201903	Control	804.400000	804.884353	2019-03-01
25	201903	Other stores	625.224335	625.224335	2019-03-01
26	201903	Trial	1026.800000	1026.800000	2019-03-01
27	201904	Control	844.600000	845.108558	2019-04-01
28	201904	Other stores	601.339544	601.339544	2019-04-01
29	201904	Trial	848.200000	848.200000	2019-04-01
30	201905	Control	922.850000	923.405675	2019-05-01
31	201905	Other stores	593.507663	593.507663	2019-05-01
32	201905	Trial	889.300000	889.300000	2019-05-01
33	201906	Control	857.200000	857.716145	2019-06-01
34	201906	Other stores	606.272519	606.272519	2019-06-01
35	201906	Trial	838.000000	838.000000	2019-06-01

In [137…

```
# Control store 95th percentile and 5th percentile
stdDev = pastSales['scaledControlSales'].std()
pastSales_Controls95 = pastSales[pastSales['Store_type'] == 'Control'].copy()
pastSales_Controls95['scaledControlSales'] = pastSales_Controls95['scaledControlSales']
pastSales_Controls95['Store_type'] = 'Control 95th % confidence interval'

pastSales_Controls5 = pastSales[pastSales['Store_type'] == 'Control'].copy()
```

```
pastSales_Controls5['scaledControlSales'] = pastSales_Controls5['scaledControlSales'] *
pastSales_Controls5['Store_type'] = 'Control 5th % confidence interval'
```

In [138…
```
# Rename columns before merging to avoid conflicts
pastSales_Controls95.rename(columns={'scaledControlSales': 'scaled95'}, inplace=True)
pastSales_Controls5.rename(columns={'scaledControlSales': 'scaled5'}, inplace=True)
```

In [139…
```
# Rename columns before merging to avoid conflicts
pastSales_Controls95.rename(columns={'scaledControlSales': 'scaled95'}, inplace=True)
pastSales_Controls5.rename(columns={'scaledControlSales': 'scaled5'}, inplace=True)
```

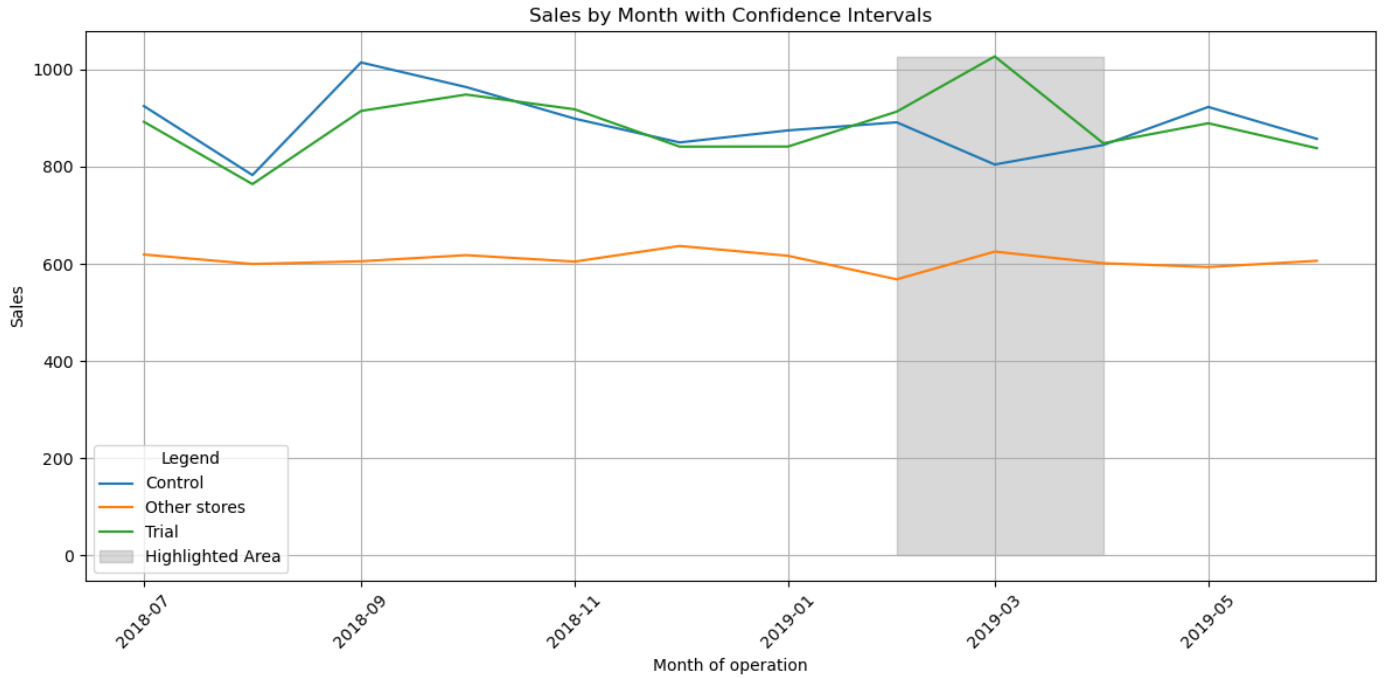
In [140…
```
#Building final df for sales assessment

combined_data_95 = pd.merge(
    pastSales,
    pastSales_Controls95[['TransactionMonth', 'scaled95']],
    on='TransactionMonth'
)


combined_data = pd.merge(
    combined_data_95,
    pastSales_Controls5[['TransactionMonth', 'scaled5']],
    on='TransactionMonth'
)


combined_data.rename(columns={'scaledControlSales': 'scaledNormal'}, inplace=True)


combined_data.drop_duplicates(inplace=True)


combined_data.head()
```

Out[140]:

	YEARMONTH	Store_type	totSales	scaledNormal	TransactionMonth	scaled95	scaled5
0	201807	Control	924.600000	925.156729	2018-07-01	266171.648788	-264321.335330
1	201807	Other stores	619.160985	619.160985	2018-07-01	266171.648788	-264321.335330
2	201807	Trial	892.200000	892.200000	2018-07-01	266171.648788	-264321.335330
3	201808	Control	782.700000	783.171286	2018-08-01	225321.814305	-223755.471732
4	201808	Other stores	599.748276	599.748276	2018-08-01	225321.814305	-223755.471732

In [141…
```
# Plotting the difference between control and trial stores on sales basis in the trial p

plt.figure(figsize=(12, 6))

sns.lineplot(data=combined_data, x='TransactionMonth', y='totSales', hue='Store_type', p


highlight_period = (combined_data['TransactionMonth'] < pd.to_datetime('2019-05-01')) &
highlight_data = combined_data[highlight_period]

plt.fill_between(
    highlight_data['TransactionMonth'],
    0,
    highlight_data['totSales'].max(),
    color='grey',
    alpha=0.3,
    label='Highlighted Area',
```

```
    zorder=1
)


plt.xlabel('Month of operation')
plt.ylabel('Sales')
plt.title('Sales by Month with Confidence Intervals')


handles, labels = plt.gca().get_legend_handles_labels()
unique_labels = []
unique_handles = []
for handle, label in zip(handles, labels):
    if label not in unique_labels:
        unique_labels.append(label)
        unique_handles.append(handle)
plt.legend(handles=unique_handles, labels=unique_labels, title='Legend')

plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()


plt.show()
```



Here we can see in this trial-control pair that the trial store has more sales than the control store in this observation period

In [142…
```
# Calculatingscaling factor for customers
scalingFactorForControlcustomers = pre_trial_period[pre_trial_period['Store_type'] == 'T
scalingFactorForControlcustomers
```

Out[142]:
```
1.033726812816189
```

In [143…
```
# Applying scaling factor to control store sales
measure_over_time['scaledControlcustomers'] = measure_over_time.apply(
    lambda row: row['nCustomers'] * scalingFactorForControlSales if row['STORE_NBR'] ==
    axis=1
)
```

In [144…
```
# Recalculating pastcustomer with scaled customer sales
```

```
pastcustomer = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'nCustomers':
pastcustomer['TransactionMonth'] = pd.to_datetime(pastcustomer['YEARMONTH'].astype(str)
pastcustomer.head()
```

Out[144]:

	YEARMONTH	Store_type	nCustomers	scaledControlcustomers	TransactionMonth
0	201807	Control	101.000000	101.060815	2018-07-01
1	201807	Other stores	70.378788	70.378788	2018-07-01
2	201807	Trial	99.000000	99.000000	2018-07-01
3	201808	Control	91.000000	91.054794	2018-08-01
4	201808	Other stores	71.007663	71.007663	2018-08-01

In [145…

```
# Converting YEARMONTH to TransactionMonth
pastcustomer['TransactionMonth'] = pd.to_datetime(pastcustomer['YEARMONTH'].astype(str)
pastcustomer.head()
```

Out[145]:

	YEARMONTH	Store_type	nCustomers	scaledControlcustomers	TransactionMonth
0	201807	Control	101.000000	101.060815	2018-07-01
1	201807	Other stores	70.378788	70.378788	2018-07-01
2	201807	Trial	99.000000	99.000000	2018-07-01
3	201808	Control	91.000000	91.054794	2018-08-01
4	201808	Other stores	71.007663	71.007663	2018-08-01

In [146…

```
# Defining highlight period
highlight_start = pd.to_datetime('2019-01-01')
highlight_end = pd.to_datetime('2019-05-01')
highlight_period = (pastcustomer['TransactionMonth'] >= highlight_start) & (pastcustomer
```

In [147…

```
# Control store 95th percentile and 5th percentile
stdDev_nc = pastcustomer[pastcustomer['Store_type'] == 'Control']['nCustomers'].std()

pastcustomer95 = pastcustomer[pastcustomer['Store_type'] == 'Control'].copy()
pastcustomer95['scaledControlcustomers'] = pastcustomer95['scaledControlcustomers'] * (1
pastcustomer95['Store_type'] = 'Control 95th % confidence interval'


pastcustomer5 = pastcustomer[pastcustomer['Store_type'] == 'Control'].copy()
pastcustomer5['scaledControlcustomers'] = pastcustomer5['scaledControlcustomers'] * (1 -
pastcustomer5['Store_type'] = 'Control 5th % confidence interval'
```

In [148…

```
# Rename columns before merging to avoid conflicts
pastcustomer95.rename(columns={'scaledControlcustomers': 'scaledcust95'}, inplace=True)
pastcustomer5.rename(columns={'scaledControlcustomers': 'scaledcust5'}, inplace=True)
```

In [149…

```
#Building final df for customer assessment

pastcustomer95 = pd.merge(
    pastcustomer,
    pastcustomer95[['TransactionMonth', 'scaledcust95']],
    on='TransactionMonth'
)


mergedata = pd.merge(
    pastcustomer95,
```

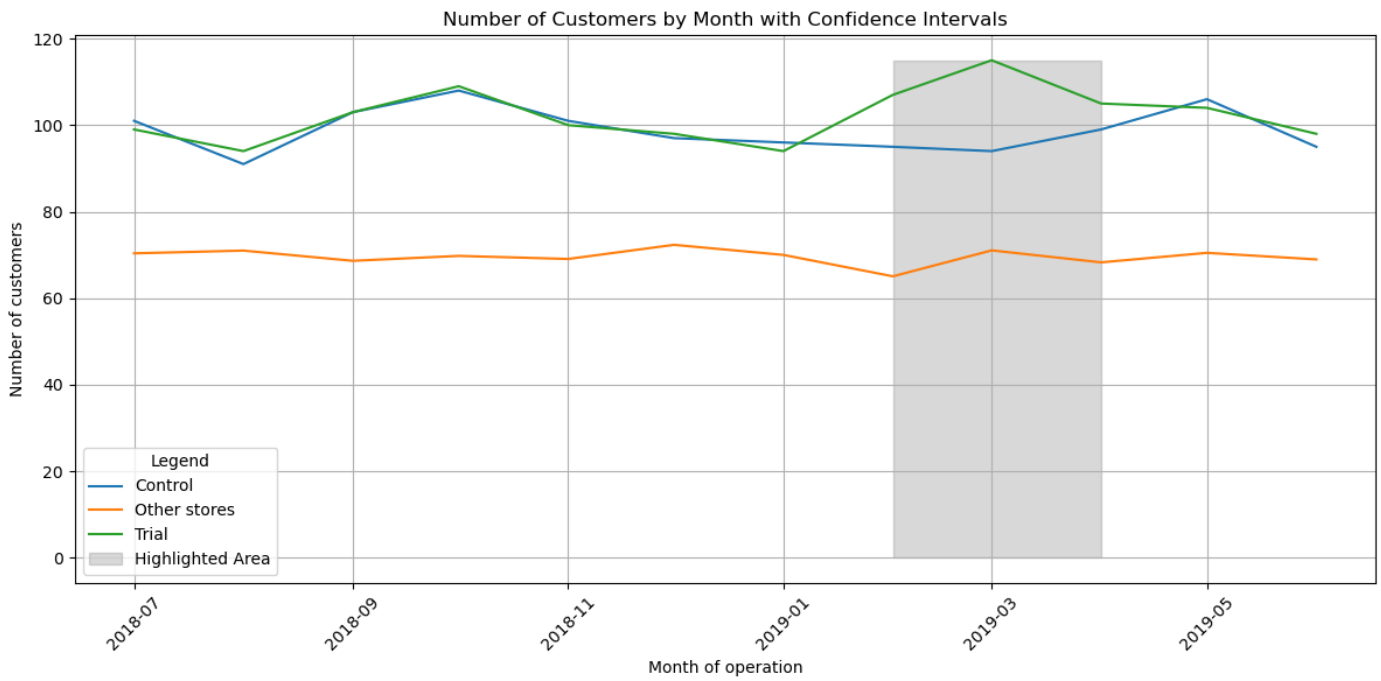
```
    pastcustomer5[['TransactionMonth', 'scaledcust5']],
    on='TransactionMonth'
)


mergedata.rename(columns={'scaledControlcustomers': 'scaledNormal'}, inplace=True)


mergedata.drop_duplicates(inplace=True)


mergedata.head()
```

Out[149]:

	YEARMONTH	Store_type	nCustomers	scaledNormal	TransactionMonth	scaledcust95	scaledcust5
0	201807	Control	101.000000	101.060815	2018-07-01	1134.681567	-932.559937
1	201807	Other stores	70.378788	70.378788	2018-07-01	1134.681567	-932.559937
2	201807	Trial	99.000000	99.000000	2018-07-01	1134.681567	-932.559937
3	201808	Control	91.000000	91.054794	2018-08-01	1022.336857	-840.227270
4	201808	Other stores	71.007663	71.007663	2018-08-01	1022.336857	-840.227270

In [150…

```
# Plotting the difference between control and trial stores on sales basis in the trial p
plt.figure(figsize=(12, 6))
sns.lineplot(data=mergedata, x='TransactionMonth', y='nCustomers', hue='Store_type', pal

highlight_period = (mergedata['TransactionMonth'] < pd.to_datetime('2019-05-01')) & (mer
highlight_data = mergedata[highlight_period]

plt.fill_between(
    highlight_data['TransactionMonth'],
    0,
    highlight_data['nCustomers'].max(),
    color='grey',
    alpha=0.3,
    label='Highlighted Area',
    zorder=1
)

plt.xlabel('Month of operation')
plt.ylabel('Number of customers')
plt.title('Number of Customers by Month with Confidence Intervals')

handles, labels = plt.gca().get_legend_handles_labels()
unique_labels = []
unique_handles = []
for handle, label in zip(handles, labels):
    if label not in unique_labels:
        unique_labels.append(label)
        unique_handles.append(handle)
plt.legend(handles=unique_handles, labels=unique_labels, title='Legend')

plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```



In this case,the trial store is better than the control store in terms of number of customers gained.

Analysing the Third Trial and control pair (88 and 237)

We will analyse on the basis of total sales and number of customers.

In [152…
```
# Set trial and control stores
trial_store = 88
control_store = 237
```

In [153…
```
# Adding Store_type based on STORE_NBR
measure_over_time['Store_type'] = measure_over_time['STORE_NBR'].apply(
    lambda x: 'Trial' if x == trial_store else ('Control' if x == control_store else 'Ot
)
```

In [154…
```
# Calculating average total sales by YEARMONTH and Store_type
pastSales = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'totSales': 'mea
pastSales
```

Out[154]:

	YEARMONTH	Store_type	totSales
0	201807	Control	1448.400000
1	201807	Other stores	615.594318
2	201807	Trial	1310.000000
3	201808	Control	1367.800000
4	201808	Other stores	595.361877
5	201808	Trial	1323.800000
6	201809	Control	1322.200000
7	201809	Other stores	602.201527
8	201809	Trial	1423.000000
9	201810	Control	1348.300000
10	201810	Other stores	614.885932
11	201810	Trial	1352.400000

12	201811	Control	1397.600000
13	201811	Other stores	600.966794
14	201811	Trial	1382.800000
15	201812	Control	1265.000000
16	201812	Other stores	633.422222
17	201812	Trial	1325.200000
18	201901	Control	1219.700000
19	201901	Other stores	613.625287
20	201901	Trial	1266.400000
21	201902	Control	1404.800000
22	201902	Other stores	564.465649
23	201902	Trial	1370.200000
24	201903	Control	1208.200000
25	201903	Other stores	621.976426
26	201903	Trial	1477.200000
27	201904	Control	1204.600000
28	201904	Other stores	597.722814
29	201904	Trial	1439.400000
30	201905	Control	1199.300000
31	201905	Other stores	590.843295
32	201905	Trial	1308.250000
33	201906	Control	1153.600000
34	201906	Other stores	603.169466
35	201906	Trial	1354.600000

In [155…

```
# Converting YEARMONTH to datetime format
pastSales['TransactionMonth'] = pd.to_datetime(pastSales['YEARMONTH'].astype(str) + '01'
pastSales['TransactionMonth']
```

Out[155]:

```
0    2018-07-01
1    2018-07-01
2    2018-07-01
3    2018-08-01
4    2018-08-01
5    2018-08-01
6    2018-09-01
7    2018-09-01
8    2018-09-01
9    2018-10-01
10   2018-10-01
11   2018-10-01
12   2018-11-01
13   2018-11-01
14   2018-11-01
15   2018-12-01
16   2018-12-01
17   2018-12-01
```

```
18   2019-01-01
19   2019-01-01
20   2019-01-01
21   2019-02-01
22   2019-02-01
23   2019-02-01
24   2019-03-01
25   2019-03-01
26   2019-03-01
27   2019-04-01
28   2019-04-01
29   2019-04-01
30   2019-05-01
31   2019-05-01
32   2019-05-01
33   2019-06-01
34   2019-06-01
35   2019-06-01
Name: TransactionMonth, dtype: datetime64[ns]
```

In [156…

```
# Calculating average number of customers by YEARMONTH and Store_type
customersacquired = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'nCustom
customersacquired
```

Out[156]:

	YEARMONTH	Store_type	nCustomers
0	201807	Control	128.000000
1	201807	Other stores	70.162879
2	201807	Trial	129.000000
3	201808	Control	135.000000
4	201808	Other stores	70.697318
5	201808	Trial	131.000000
6	201809	Control	126.000000
7	201809	Other stores	68.477099
8	201809	Trial	124.000000
9	201810	Control	123.000000
10	201810	Other stores	69.673004
11	201810	Trial	123.000000
12	201811	Control	132.000000
13	201811	Other stores	68.843511
14	201811	Trial	130.000000
15	201812	Control	124.000000
16	201812	Other stores	72.130268
17	201812	Trial	126.000000
18	201901	Control	117.000000
19	201901	Other stores	69.842912
20	201901	Trial	117.000000
21	201902	Control	126.000000
22	201902	Other stores	64.881679

23	201902	Trial	124.000000
24	201903	Control	119.000000
25	201903	Other stores	70.889734
26	201903	Trial	134.000000
27	201904	Control	120.000000
28	201904	Other stores	68.121673
29	201904	Trial	128.000000
30	201905	Control	129.000000
31	201905	Other stores	70.310345
32	201905	Trial	128.000000
33	201906	Control	119.000000
34	201906	Other stores	68.793893
35	201906	Trial	121.000000

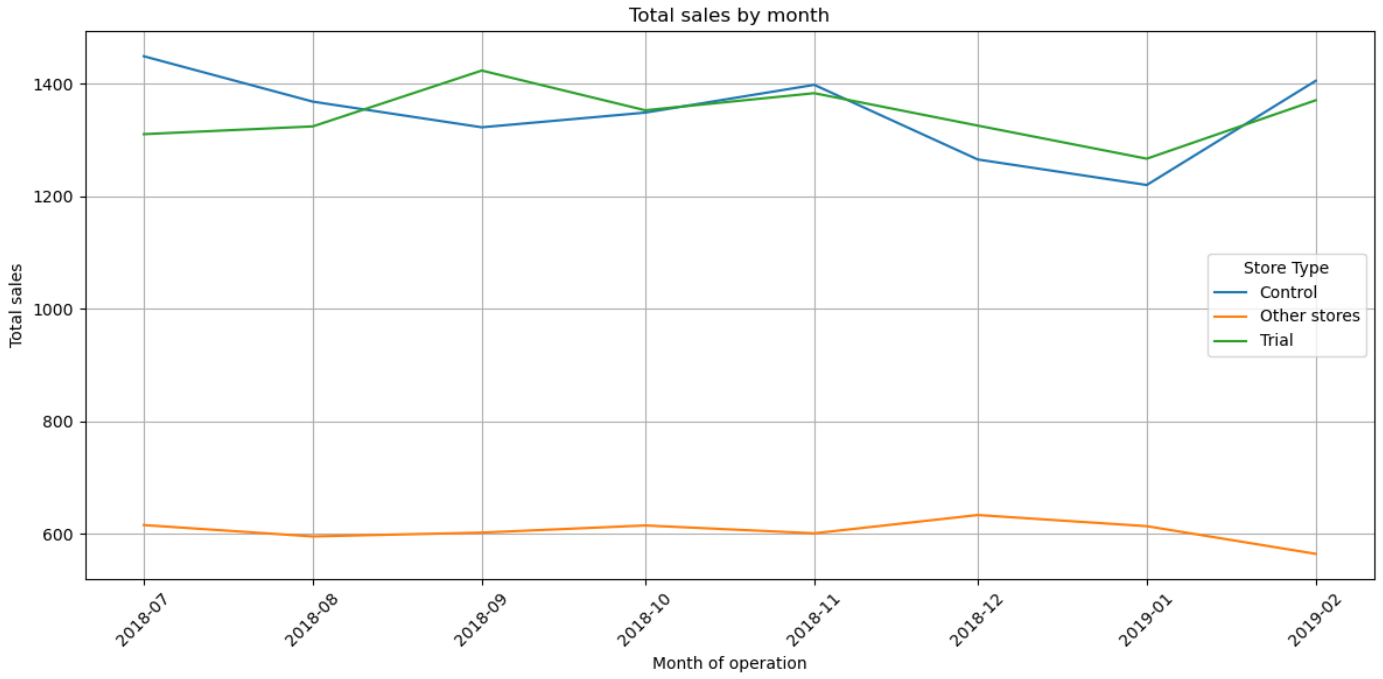
In [157…

```
# Converting YEARMONTH to datetime format
customersacquired['AcquireMonth'] = pd.to_datetime(customersacquired['YEARMONTH'].astype
customersacquired['AcquireMonth']
```

Out[157]:

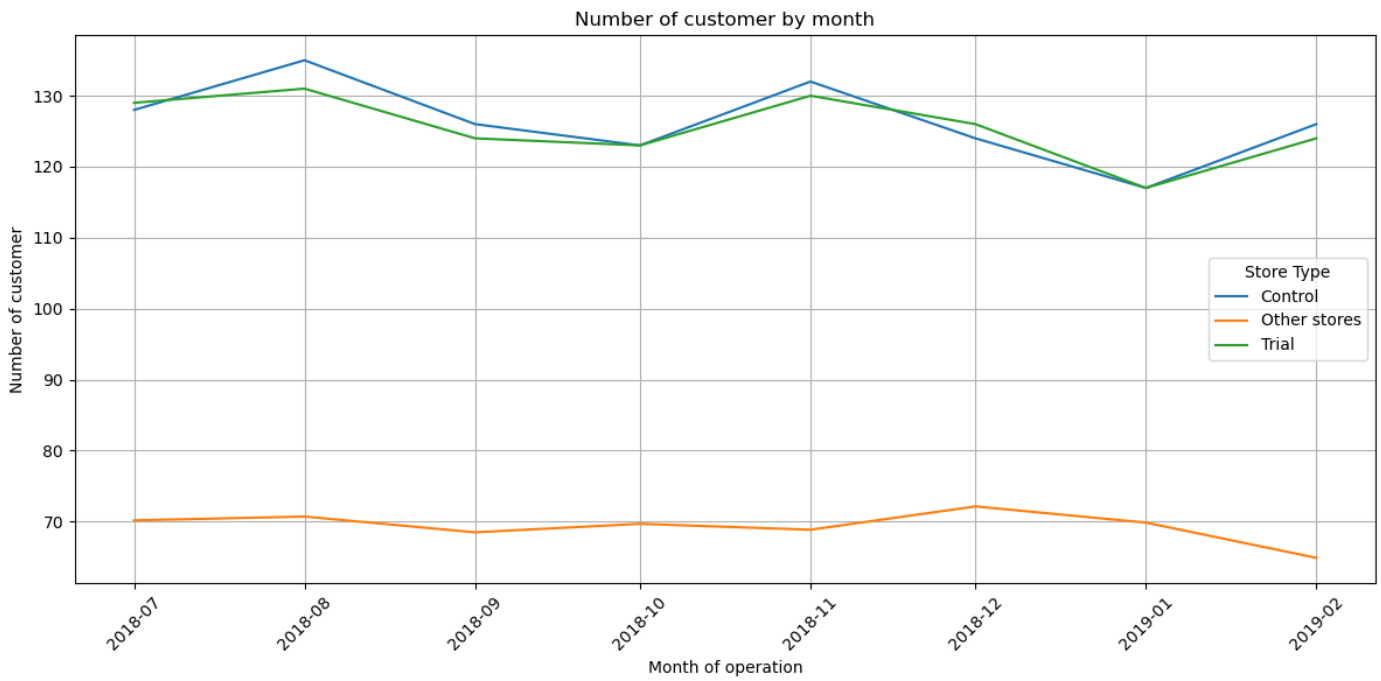
```
0    2018-07-01
1    2018-07-01
2    2018-07-01
3    2018-08-01
4    2018-08-01
5    2018-08-01
6    2018-09-01
7    2018-09-01
8    2018-09-01
9    2018-10-01
10   2018-10-01
11   2018-10-01
12   2018-11-01
13   2018-11-01
14   2018-11-01
15   2018-12-01
16   2018-12-01
17   2018-12-01
18   2019-01-01
19   2019-01-01
20   2019-01-01
21   2019-02-01
22   2019-02-01
23   2019-02-01
24   2019-03-01
25   2019-03-01
26   2019-03-01
27   2019-04-01
28   2019-04-01
29   2019-04-01
30   2019-05-01
31   2019-05-01
32   2019-05-01
33   2019-06-01
34   2019-06-01
35   2019-06-01
Name: AcquireMonth, dtype: datetime64[ns]
```

In [158…
```
# Plot the data on basis of total sales by month

pastSales = pastSales[pastSales['YEARMONTH'] < 201903]


plt.figure(figsize=(12, 6))
sns.lineplot(data=pastSales, x='TransactionMonth', y='totSales', hue='Store_type')
plt.xlabel('Month of operation')
plt.ylabel('Total sales')
plt.title('Total sales by month')
plt.legend(title='Store Type')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Here we can see that the trial store has higher sales than the control store before the Trial period.

In [159…
```
# Plotting the data on basis of number of customers by month

customersacquired = customersacquired[customersacquired['YEARMONTH'] < 201903]


plt.figure(figsize=(12, 6))
sns.lineplot(data=customersacquired, x='AcquireMonth', y='nCustomers', hue='Store_type')
plt.xlabel('Month of operation')
plt.ylabel('Number of customer')
plt.title('Number of customer by month')
plt.legend(title='Store Type')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Both type of stores have almost the same number of customers acquired per week.

In [161…
```
# Sampling pre-trial measures for demonstration
pre_trial_period = measure_over_time[(measure_over_time['STORE_NBR'] == trial_store) |
                                     (measure_over_time['STORE_NBR'] == control_store
pre_trial_period
```

Out[161]:

	STORE_NBR	YEARMONTH	totSales	nCustomers	nTxn	totQty	nTxnPerCust	nChipsPerTxn	avgPricePerUn
1001	88	201807	1310.00	129	153	306	1.186047	2.000000	4.28104
1002	88	201808	1323.80	131	158	303	1.206107	1.917722	4.36897
1003	88	201809	1423.00	124	157	318	1.266129	2.025478	4.47484
1004	88	201810	1352.40	123	155	316	1.260163	2.038710	4.27974
1005	88	201811	1382.80	130	156	314	1.200000	2.012821	4.40382
1006	88	201812	1325.20	126	148	298	1.174603	2.013514	4.44698
1007	88	201901	1266.40	117	144	292	1.230769	2.027778	4.33698
1008	88	201902	1370.20	124	153	308	1.233871	2.013072	4.44870
1009	88	201903	1477.20	134	169	340	1.261194	2.011834	4.34470
1010	88	201904	1439.40	128	162	324	1.265625	2.000000	4.44259
1011	88	201905	1308.25	128	154	299	1.203125	1.941558	4.37541
1012	88	201906	1354.60	121	148	300	1.223140	2.027027	4.51533
2747	237	201807	1448.40	128	162	324	1.265625	2.000000	4.47037
2748	237	201808	1367.80	135	165	313	1.222222	1.896970	4.36996
2749	237	201809	1322.20	126	149	299	1.182540	2.006711	4.42207
2750	237	201810	1348.30	123	147	299	1.195122	2.034014	4.50936
2751	237	201811	1397.60	132	161	320	1.219697	1.987578	4.36750
2752	237	201812	1265.00	124	144	289	1.161290	2.006944	4.37716
2753	237	201901	1219.70	117	139	277	1.188034	1.992806	4.40324

2754	237	201902	1404.80	126	157	314	1.246032	2.000000	4.47388
2755	237	201903	1208.20	119	134	274	1.126050	2.044776	4.40948
2756	237	201904	1204.60	120	135	272	1.125000	2.014815	4.42867
2757	237	201905	1199.30	129	149	272	1.155039	1.825503	4.40919
2758	237	201906	1153.60	119	131	262	1.100840	2.000000	4.40305

In [162…
```
# Calculate scaling factor for sales
scalingFactorForControlSales = pre_trial_period[pre_trial_period['Store_type'] == 'Trial
scalingFactorForControlSales
```

Out[162]:
```
1.0510795070626469
```

In [163…
```
# Apply scaling factor to control store sales
measure_over_time['scaledControlSales'] = measure_over_time.apply(
    lambda row: row['totSales'] * scalingFactorForControlSales if row['STORE_NBR'] == co
    axis=1
)
```

In [164…
```
# Recalculating pastSales with scaled control sales
pastSales = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'totSales': 'mea
pastSales['TransactionMonth'] = pd.to_datetime(pastSales['YEARMONTH'].astype(str) + '01'
pastSales
```

Out[164]:

	YEARMONTH	Store_type	totSales	scaledControlSales	TransactionMonth
0	201807	Control	1448.400000	1522.383558	2018-07-01
1	201807	Other stores	615.594318	615.594318	2018-07-01
2	201807	Trial	1310.000000	1310.000000	2018-07-01
3	201808	Control	1367.800000	1437.666550	2018-08-01
4	201808	Other stores	595.361877	595.361877	2018-08-01
5	201808	Trial	1323.800000	1323.800000	2018-08-01
6	201809	Control	1322.200000	1389.737324	2018-09-01
7	201809	Other stores	602.201527	602.201527	2018-09-01
8	201809	Trial	1423.000000	1423.000000	2018-09-01
9	201810	Control	1348.300000	1417.170499	2018-10-01
10	201810	Other stores	614.885932	614.885932	2018-10-01
11	201810	Trial	1352.400000	1352.400000	2018-10-01
12	201811	Control	1397.600000	1468.988719	2018-11-01
13	201811	Other stores	600.966794	600.966794	2018-11-01
14	201811	Trial	1382.800000	1382.800000	2018-11-01
15	201812	Control	1265.000000	1329.615576	2018-12-01
16	201812	Other stores	633.422222	633.422222	2018-12-01
17	201812	Trial	1325.200000	1325.200000	2018-12-01
18	201901	Control	1219.700000	1282.001675	2019-01-01
19	201901	Other stores	613.625287	613.625287	2019-01-01
20	201901	Trial	1266.400000	1266.400000	2019-01-01

21	201902	Control	1404.800000	1476.556492	2019-02-01
22	201902	Other stores	564.465649	564.465649	2019-02-01
23	201902	Trial	1370.200000	1370.200000	2019-02-01
24	201903	Control	1208.200000	1269.914260	2019-03-01
25	201903	Other stores	621.976426	621.976426	2019-03-01
26	201903	Trial	1477.200000	1477.200000	2019-03-01
27	201904	Control	1204.600000	1266.130374	2019-04-01
28	201904	Other stores	597.722814	597.722814	2019-04-01
29	201904	Trial	1439.400000	1439.400000	2019-04-01
30	201905	Control	1199.300000	1260.559653	2019-05-01
31	201905	Other stores	590.843295	590.843295	2019-05-01
32	201905	Trial	1308.250000	1308.250000	2019-05-01
33	201906	Control	1153.600000	1212.525319	2019-06-01
34	201906	Other stores	603.169466	603.169466	2019-06-01
35	201906	Trial	1354.600000	1354.600000	2019-06-01

In [165…
```
# Control store 95th percentile and 5th percentile
stdDev = pastSales['scaledControlSales'].std()
pastSales_Controls95 = pastSales[pastSales['Store_type'] == 'Control'].copy()
pastSales_Controls95['scaledControlSales'] = pastSales_Controls95['scaledControlSales']
pastSales_Controls95['Store_type'] = 'Control 95th % confidence interval'

pastSales_Controls5 = pastSales[pastSales['Store_type'] == 'Control'].copy()
pastSales_Controls5['scaledControlSales'] = pastSales_Controls5['scaledControlSales'] *
pastSales_Controls5['Store_type'] = 'Control 5th % confidence interval'
```

In [166…
```
# Rename columns before merging to avoid conflicts
pastSales_Controls95.rename(columns={'scaledControlSales': 'scaled95'}, inplace=True)
pastSales_Controls5.rename(columns={'scaledControlSales': 'scaled5'}, inplace=True)
```

In [167…
```
#Building final df for sales assessment

combined_data_95 = pd.merge(
    pastSales,
    pastSales_Controls95[['TransactionMonth', 'scaled95']],
    on='TransactionMonth'
)


combined_data = pd.merge(
    combined_data_95,
    pastSales_Controls5[['TransactionMonth', 'scaled5']],
    on='TransactionMonth'
)


combined_data.rename(columns={'scaledControlSales': 'scaledNormal'}, inplace=True)


combined_data.drop_duplicates(inplace=True)
```

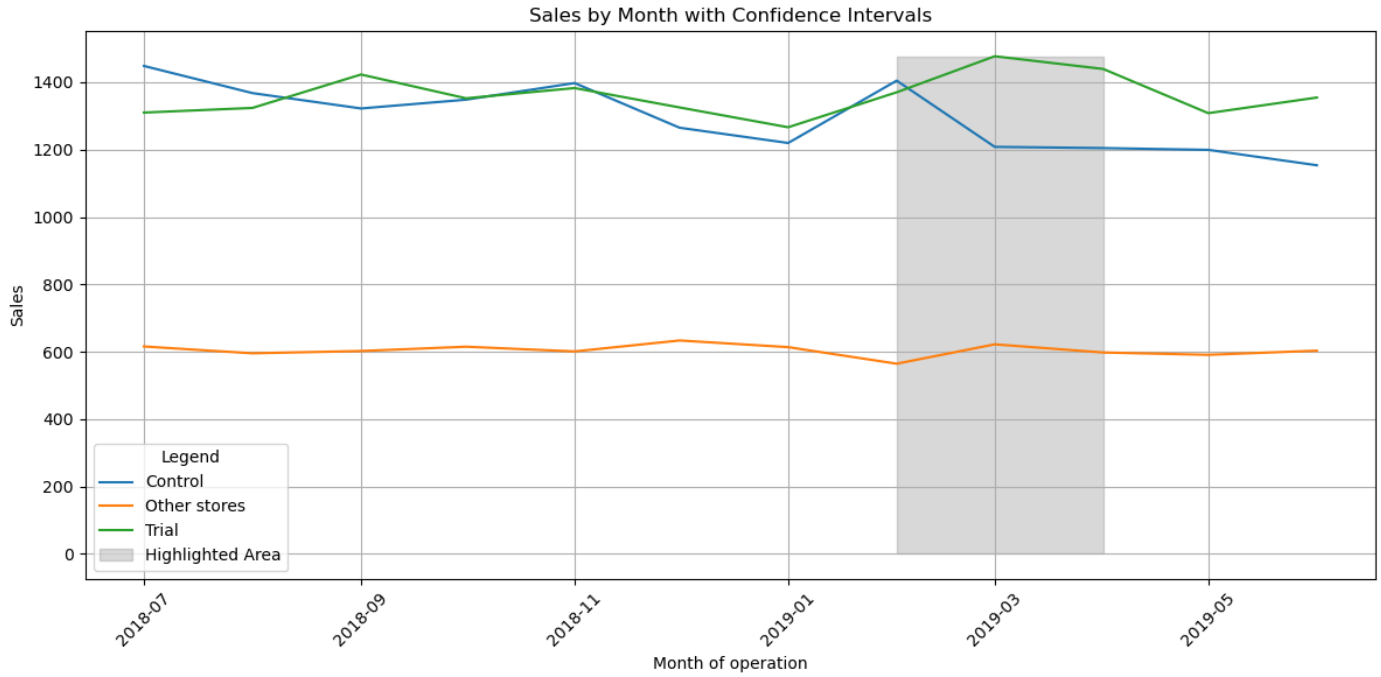
```
combined_data.head()
```

Out[167]:

	YEARMONTH	Store_type	totSales	scaledNormal	TransactionMonth	scaled95	scaled5
0	201807	Control	1448.400000	1522.383558	2018-07-01	1.122192e+06	-1.119148e+06
1	201807	Other stores	615.594318	615.594318	2018-07-01	1.122192e+06	-1.119148e+06
2	201807	Trial	1310.000000	1310.000000	2018-07-01	1.122192e+06	-1.119148e+06
3	201808	Control	1367.800000	1437.666550	2018-08-01	1.059745e+06	-1.056870e+06
4	201808	Other stores	595.361877	595.361877	2018-08-01	1.059745e+06	-1.056870e+06

In [168…

```
# Plotting the difference between control and trial stores on sales basis in the trial p

plt.figure(figsize=(12, 6))

sns.lineplot(data=combined_data, x='TransactionMonth', y='totSales', hue='Store_type', p


highlight_period = (combined_data['TransactionMonth'] < pd.to_datetime('2019-05-01')) &
highlight_data = combined_data[highlight_period]

plt.fill_between(
    highlight_data['TransactionMonth'],
    0,
    highlight_data['totSales'].max(),
    color='grey',
    alpha=0.3,
    label='Highlighted Area',
    zorder=1
)


plt.xlabel('Month of operation')
plt.ylabel('Sales')
plt.title('Sales by Month with Confidence Intervals')


handles, labels = plt.gca().get_legend_handles_labels()
unique_labels = []
unique_handles = []
for handle, label in zip(handles, labels):
    if label not in unique_labels:
        unique_labels.append(label)
        unique_handles.append(handle)
plt.legend(handles=unique_handles, labels=unique_labels, title='Legend')

plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()


plt.show()
```



In this case, we see during the observation period that the Trial store gets more revenue than the control store.

In [169…
```
# Calculate scaling factor for customers
scalingFactorForControlcustomers = pre_trial_period[pre_trial_period['Store_type'] == 'T
scalingFactorForControlcustomers
```

Out[169]:
```
1.0113484646194926
```

In [170…
```
# Applying scaling factor to control store sales
measure_over_time['scaledControlcustomers'] = measure_over_time.apply(
    lambda row: row['nCustomers'] * scalingFactorForControlSales if row['STORE_NBR'] ==
    axis=1
)
```

In [171…
```
# Recalculating pastcustomer with scaled customer sales
pastcustomer = measure_over_time.groupby(['YEARMONTH', 'Store_type']).agg({'nCustomers':
pastcustomer['TransactionMonth'] = pd.to_datetime(pastcustomer['YEARMONTH'].astype(str)
pastcustomer.head()
```

Out[171]:

	YEARMONTH	Store_type	nCustomers	scaledControlcustomers	TransactionMonth
0	201807	Control	128.000000	134.538177	2018-07-01
1	201807	Other stores	70.162879	70.162879	2018-07-01
2	201807	Trial	129.000000	129.000000	2018-07-01
3	201808	Control	135.000000	141.895733	2018-08-01
4	201808	Other stores	70.697318	70.697318	2018-08-01

In [172…
```
# Converting YEARMONTH to TransactionMonth
pastcustomer['TransactionMonth'] = pd.to_datetime(pastcustomer['YEARMONTH'].astype(str)
pastcustomer.head()
```

Out[172]:

	YEARMONTH	Store_type	nCustomers	scaledControlcustomers	TransactionMonth
0	201807	Control	128.000000	134.538177	2018-07-01
1	201807	Other stores	70.162879	70.162879	2018-07-01
2	201807	Trial	129.000000	129.000000	2018-07-01

3	201808	Control	135.000000	141.895733	2018-08-01
4	201808	Other stores	70.697318	70.697318	2018-08-01

In [173…
```
# Defining highlight period
highlight_start = pd.to_datetime('2019-01-01')
highlight_end = pd.to_datetime('2019-05-01')
highlight_period = (pastcustomer['TransactionMonth'] >= highlight_start) & (pastcustomer
```

In [174…
```
# Control store 95th percentile and 5th percentile
stdDev_nc = pastcustomer[pastcustomer['Store_type'] == 'Control']['nCustomers'].std()

pastcustomer95 = pastcustomer[pastcustomer['Store_type'] == 'Control'].copy()
pastcustomer95['scaledControlcustomers'] = pastcustomer95['scaledControlcustomers'] * (1
pastcustomer95['Store_type'] = 'Control 95th % confidence interval'


pastcustomer5 = pastcustomer[pastcustomer['Store_type'] == 'Control'].copy()
pastcustomer5['scaledControlcustomers'] = pastcustomer5['scaledControlcustomers'] * (1 -
pastcustomer5['Store_type'] = 'Control 5th % confidence interval'
```

In [175…
```
# Rename columns before merging to avoid conflicts
pastcustomer95.rename(columns={'scaledControlcustomers': 'scaledcust95'}, inplace=True)
pastcustomer5.rename(columns={'scaledControlcustomers': 'scaledcust5'}, inplace=True)
```

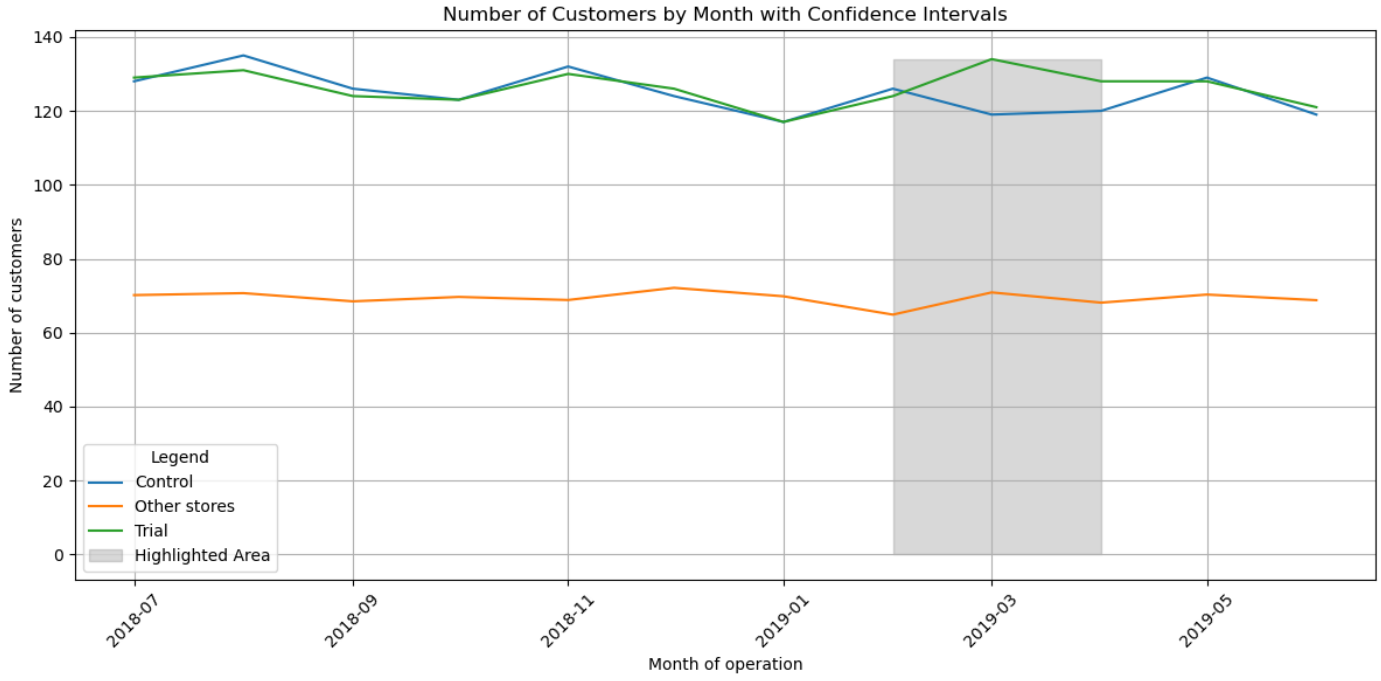
In [176…
```
#Building final df for customer assessment

pastcustomer95 = pd.merge(
    pastcustomer,
    pastcustomer95[['TransactionMonth', 'scaledcust95']],
    on='TransactionMonth'
)


mergedata = pd.merge(
    pastcustomer95,
    pastcustomer5[['TransactionMonth', 'scaledcust5']],
    on='TransactionMonth'
)


mergedata.rename(columns={'scaledControlcustomers': 'scaledNormal'}, inplace=True)


mergedata.drop_duplicates(inplace=True)


mergedata.head()
```

Out[176]:

	YEARMONTH	Store_type	nCustomers	scaledNormal	TransactionMonth	scaledcust95	scaledcust5
0	201807	Control	128.000000	134.538177	2018-07-01	1634.155666	-1365.079313
1	201807	Other stores	70.162879	70.162879	2018-07-01	1634.155666	-1365.079313
2	201807	Trial	129.000000	129.000000	2018-07-01	1634.155666	-1365.079313
3	201808	Control	135.000000	141.895733	2018-08-01	1723.523554	-1439.732087
4	201808	Other stores	70.697318	70.697318	2018-08-01	1723.523554	-1439.732087

In [177…
```
# Plotting the difference between control and trial stores on sales basis in the trial p
plt.figure(figsize=(12, 6))
```

```
sns.lineplot(data=mergedata, x='TransactionMonth', y='nCustomers', hue='Store_type', pal

highlight_period = (mergedata['TransactionMonth'] < pd.to_datetime('2019-05-01')) & (mer
highlight_data = mergedata[highlight_period]

plt.fill_between(
    highlight_data['TransactionMonth'],
    0,
    highlight_data['nCustomers'].max(),
    color='grey',
    alpha=0.3,
    label='Highlighted Area',
    zorder=1
)

plt.xlabel('Month of operation')
plt.ylabel('Number of customers')
plt.title('Number of Customers by Month with Confidence Intervals')

handles, labels = plt.gca().get_legend_handles_labels()
unique_labels = []
unique_handles = []
for handle, label in zip(handles, labels):
    if label not in unique_labels:
        unique_labels.append(label)
        unique_handles.append(handle)
plt.legend(handles=unique_handles, labels=unique_labels, title='Legend')

plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```



Same case as above that trial stores have gained a higher number of customers than the control stores.