In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [ ]:
```

```

In [2]:
```
client_df = pd.read_csv("datasets/module_two_datasets/QVI_data.csv")
```

Select control stores

The client has selected store numbers 77, 86 and 88 as trial stores and want control stores to be established stores that are operational for the entire observation period.

We would want to match trial stores to control stores that are similar to the trial store prior to the trial period of Feb 2019 in terms of :

- Monthly overall sales revenue
- Monthly number of customers
- Monthly number of transactions per customer

In [3]:
```
client_df.head()
```

Out[3]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_S
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 [4]:
```
# check for null values
client_df.isnull().sum()
```

Out[4]:
```
LYLTY_CARD_NBR      0
DATE                0
STORE_NBR           0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
PACK_SIZE           0
BRAND               0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [5]:
```
# check datatypes
client_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 [6]:
```
# change date column to datetime format
client_df['DATE'] = pd.to_datetime(client_df['DATE'])
```

In [7]:
```
1 client_df.head()
```

Out[7]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_S
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 [8]:
```
1 client_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  datetime64[ns]
 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: datetime64[ns](1), float64(1), int64(6), object(4)
memory usage: 24.2+ MB
```

In [9]:
```
# check unique values for all features
client_df.nunique()
```

Out[9]:
```
LYLTY_CARD_NBR       72636
DATE                   364
STORE_NBR              272
TXN_ID              263125
PROD_NBR               114
PROD_NAME              114
PROD_QTY                 5
TOT_SALES              111
PACK_SIZE               21
BRAND                   21
LIFESTAGE                7
PREMIUM_CUSTOMER         3
dtype: int64
```

Client wanted to select store numbers 77, 86 88 as trial stores and want control stores to be established stores that are operational for the entire observation period. We would want to match trial stores to control stores that are similar to the trial store prior to the trial period of Feb 2019 in terms of :

- Monthly overall sales revenue
- Monthly number of customers
- Monthly number of transactions per customer

In [10]:
```
# change datatypes first
client_df['DATE'] = client_df['DATE'].astype("str")
```

In [11]:
```
# add a new month ID column with the format yyyymm
yearmonth_values = []
for i in range(client_df.shape[0]):
    # get the year data. covert to string for joining later
    client_year = client_df['DATE'][i].split("-")[0]
    # get the month data and convert to string for joining
    client_month = client_df['DATE'][i].split("-")[1]
    # add values together
    some_value = client_year+client_month
    # append to list
    yearmonth_values.append(some_value)
```

In [12]:
```
client_df['yearmonth'] = yearmonth_values
client_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264834 entries, 0 to 264833
Data columns (total 13 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 
 12  yearmonth         264834 non-null  object 
dtypes: float64(1), int64(6), object(6)
memory usage: 26.3+ MB
```

In [13]:
```
# convert month_id column to integer datatype
client_df['yearmonth'] = pd.to_numeric(client_df['yearmonth'])
```

In [14]:
```
# convert the date column back to datetime object
client_df['DATE'] = pd.to_datetime(client_df['DATE'])
```

In [15]:
```
# check if changes are made
client_df
```

Out[15]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
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
...	...	...	...	...	...	...	...
264829	2370701	2018-12-08	88	240378	24	Grain Waves Sweet Chilli 210g	2
264830	2370751	2018-10-01	88	240394	60	Kettle Tortilla ChpsFeta&Garlic 150g	2
264831	2370961	2018-10-24	88	240480	70	Tyrrells Crisps Lightly Salted 165g	2
264832	2370961	2018-10-27	88	240481	65	Old El Paso Salsa Dip Chnky Tom Ht300g	2
264833	2373711	2018-12-14	88	241815	16	Smiths Crinkle Chips Salt & Vinegar 330g	2

264834 rows × 13 columns

In [16]:
```
1 client_df.dtypes
```

Out[16]:
```
LYLTY_CARD_NBR               int64
DATE                datetime64[ns]
STORE_NBR                    int64
TXN_ID                       int64
PROD_NBR                     int64
PROD_NAME                   object
PROD_QTY                     int64
TOT_SALES                  float64
PACK_SIZE                    int64
BRAND                       object
LIFESTAGE                   object
PREMIUM_CUSTOMER            object
yearmonth                    int64
dtype: object
```

In [17]:
```
1 # sort client_df by date from lowest to highest
2 client_df = client_df.sort_values(by='DATE').reset_index(drop=True)
```

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

In [18]:
```
1 # total sales per store per date
2 total_sales = client_df.groupby(by=['STORE_NBR','yearmonth']).sum()['TOT_SAL
3 total_sales
```

Out[18]:
```
STORE_NBR  yearmonth
1          201807       206.9
           201808       176.1
           201809       278.8
           201810       188.1
           201811       192.6
                           ...
272        201902       395.5
           201903       442.3
           201904       445.1
           201905       314.6
           201906       312.1
Name: TOT_SALES, Length: 3169, dtype: float64
```

In [19]:
```
# get the number of transactions
transactions = client_df.groupby(by=['STORE_NBR','yearmonth']).count()['LYLT
transactions
```

Out[19]:
```
STORE_NBR  yearmonth
1          201807       52
           201808       43
           201809       62
           201810       45
           201811       47
                        ..
272        201902       48
           201903       53
           201904       56
           201905       40
           201906       37
Name: LYLTY_CARD_NBR, Length: 3169, dtype: int64
```

In [20]:
```
# get the number of customers
unique_customers = client_df.groupby(by=['STORE_NBR','yearmonth']).nunique()
unique_customers
```

Out[20]:
```
STORE_NBR  yearmonth
1          201807       49
           201808       42
           201809       59
           201810       44
           201811       46
                        ..
272        201902       45
           201903       50
           201904       54
           201905       34
           201906       34
Name: LYLTY_CARD_NBR, Length: 3169, dtype: int64
```

In [21]:
```
# get the number of transactions PER customer
transactions_per_customer = transactions / unique_customers
transactions_per_customer
```

Out[21]:
```
STORE_NBR  yearmonth
1          201807       1.061224
           201808       1.023810
           201809       1.050847
           201810       1.022727
           201811       1.021739
                           ...
272        201902       1.066667
           201903       1.060000
           201904       1.037037
           201905       1.176471
           201906       1.088235
Name: LYLTY_CARD_NBR, Length: 3169, dtype: float64
```

In [22]:
```
# get the number of chips purchased
chips_purchased = client_df.groupby(by=['STORE_NBR','yearmonth']).sum()['PRO
chips_purchased
```

Out[22]:
```
STORE_NBR  yearmonth
1          201807        62
           201808        54
           201809        75
           201810        58
           201811        57
                        ...
272        201902        91
           201903       101
           201904       105
           201905        71
           201906        70
Name: PROD_QTY, Length: 3169, dtype: int64
```

In [23]:
```
# get chips PER customer
chips_per_customer = chips_purchased / unique_customers
chips_per_customer
```

Out[23]:
```
STORE_NBR  yearmonth
1          201807       1.265306
           201808       1.285714
           201809       1.271186
           201810       1.318182
           201811       1.239130
                           ...
272        201902       2.022222
           201903       2.020000
           201904       1.944444
           201905       2.088235
           201906       2.058824
Length: 3169, dtype: float64
```

In [24]:
```
# get chips PER transaction
chips_per_transaction = chips_purchased / transactions
chips_per_transaction
```

Out[24]:
```
STORE_NBR  yearmonth
1          201807       1.192308
           201808       1.255814
           201809       1.209677
           201810       1.288889
           201811       1.212766
                           ...
272        201902       1.895833
           201903       1.905660
           201904       1.875000
           201905       1.775000
           201906       1.891892
Length: 3169, dtype: float64
```

In [25]:
```
# get the average price per unit
price_per_unit = total_sales / chips_purchased
price_per_unit
```

Out[25]:
```
STORE_NBR  yearmonth
1          201807       3.337097
           201808       3.261111
           201809       3.717333
           201810       3.243103
           201811       3.378947
                           ...
272        201902       4.346154
           201903       4.379208
           201904       4.239048
           201905       4.430986
           201906       4.458571
Length: 3169, dtype: float64
```

In [26]:
```
# concatenate total sales, num_customers, transactions/customer, chips/custo
items_tojoin = [total_sales, \
                unique_customers, \
                transactions_per_customer, \
#                 chips_per_customer,\
                chips_per_transaction, \
                price_per_unit]
# concatenate and store in a variable
measureOverTime_df = pd.concat(items_tojoin, axis=1)
```

In [27]:
```
measureOverTime_df.columns.values
```

Out[27]:
```
array(['TOT_SALES', 'LYLTY_CARD_NBR', 'LYLTY_CARD_NBR', 0, 1],
      dtype=object)
```

In [28]:
```
# rename columms
new_names = ["totSales", 'nCustomers','nTxnPerCust','nChipsPerTxn','avgPrice

measureOverTime_df.columns = new_names
```

In [29]:
```
# check if changes are made
measureOverTime_df.head()
```

Out[29]:

		totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
STORE_NBR	**yearmonth**					
1	**201807**	206.9	49	1.061224	1.192308	3.337097
	201808	176.1	42	1.023810	1.255814	3.261111
	201809	278.8	59	1.050847	1.209677	3.717333
	201810	188.1	44	1.022727	1.288889	3.243103
	201811	192.6	46	1.021739	1.212766	3.378947

In [30]:
```
measureOverTime_df.reset_index(inplace=True)
```

In [31]:
```
measureOverTime_df.head()
```

Out[31]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	1.061224	1.192308	3.337097
1	1	201808	176.1	42	1.023810	1.255814	3.261111
2	1	201809	278.8	59	1.050847	1.209677	3.717333
3	1	201810	188.1	44	1.022727	1.288889	3.243103
4	1	201811	192.6	46	1.021739	1.212766	3.378947

In [32]:
```
## find the stores with full observation (store with all dates)

# first find how many oberservations there are in total:
client_df['yearmonth'].nunique()
```

Out[32]: 12

In [33]:
```
# stores that don't have 12 unique "yearmonth" values don't have full observ
store_yearmonth = pd.DataFrame(measureOverTime_df.groupby(by='STORE_NBR').nu
store_yearmonth.loc[store_yearmonth['yearmonth'] != 12]
```

Out[33]:

STORE_NBR	yearmonth
11	2
31	2
44	11
76	1
85	1
92	1
117	11
193	3
206	2
211	2
218	11
252	2

In [34]:
```
# find stores that don't have 12 obervations
store_missingValues = store_yearmonth.loc[store_yearmonth['yearmonth'] != 12
```

In [35]:
```
1 store_missingValues
```

Out[35]:
```
array([ 11,  31,  44,  76,  85,  92, 117, 193, 206, 211, 218, 252],
      dtype=int64)
```

In [560]:
```
1 print(measureOverTime_df['yearmonth'].min())
2 measureOverTime_df['yearmonth'].max()
```

```
201807
```

Out[560]:
```
201906
```

In [36]:
```
1 measureOverTime_df.head()
```

Out[36]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	1.061224	1.192308	3.337097
1	1	201808	176.1	42	1.023810	1.255814	3.261111
2	1	201809	278.8	59	1.050847	1.209677	3.717333
3	1	201810	188.1	44	1.022727	1.288889	3.243103
4	1	201811	192.6	46	1.021739	1.212766	3.378947

In [37]:
```
1 # drop stores by only selecting stores that have full observations
2 storesWithFullObs = measureOverTime_df.loc[~measureOverTime_df['STORE_NBR'].
3
```

In [38]:
```
1 print(storesWithFullObs.shape)
2 storesWithFullObs.head()
```

```
(3120, 7)
```

Out[38]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	1.061224	1.192308	3.337097
1	1	201808	176.1	42	1.023810	1.255814	3.261111
2	1	201809	278.8	59	1.050847	1.209677	3.717333
3	1	201810	188.1	44	1.022727	1.288889	3.243103
4	1	201811	192.6	46	1.021739	1.212766	3.378947

In [40]:
```
# filter the pre-trial period (before february of 2019)
preTrialMeasures = storesWithFullObs.loc[storesWithFullObs['yearmonth'] < 20
preTrialMeasures.head()
```

Out[40]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	1.061224	1.192308	3.337097
1	1	201808	176.1	42	1.023810	1.255814	3.261111
2	1	201809	278.8	59	1.050847	1.209677	3.717333
3	1	201810	188.1	44	1.022727	1.288889	3.243103
4	1	201811	192.6	46	1.021739	1.212766	3.378947

In [562]:
```
# save file as html
preTrialMeasures.to_html("html_formatted_dataframes/module_two_files/preTria
```

Create a functions that calculates the **correlation** between trial store and other stores based on a single metric

In [41]:
```
client_df.head()
```

Out[41]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_S
0	104039	2018-07-01	104	103937	53	RRD Sweet Chilli & Sour Cream 165g	2	
1	118107	2018-07-01	118	121300	33	Cobs Popd Swt/Chlli &Sr/Cream Chips 110g	2	
2	226024	2018-07-01	226	226348	114	Kettle Sensations Siracha Lime 150g	2	
3	152040	2018-07-01	152	150459	25	Pringles SourCream Onion 134g	2	
4	33140	2018-07-01	33	30342	10	RRD SR Slow Rst Pork Belly 150g	2	

In [92]:
```
# create a functions that calculates the correlation between trial store and
test_df = pd.DataFrame(preTrialMeasures.loc[preTrialMeasures['STORE_NBR'] ==
pd.DataFrame(preTrialMeasures.loc[preTrialMeasures['STORE_NBR'] == 77,"totSa
```

Out[92]: 0.07521784241886638

In [124]:

```
# create a functions that calculates the correlation between trial store and
def findCorrelation(pre_trial_store, trial_store_num, measuring_metric):
    # create empty dataframe with column names to store values later
    some_dataframe = pd.DataFrame(columns =
                                  ['trial_store',"other_stores","measure","co

    # define number of stores for looping
    number_of_stores = pre_trial_store['STORE_NBR'].unique()

    for i in number_of_stores:

        # create a dataframe for the store used for comparison with just the
        df_forTrial = pd.DataFrame(pre_trial_store.loc[pre_trial_store['STOR

        # Create a dataframe for the pre-trial store with the measuring metr
        measuring_df = pd.DataFrame(pre_trial_store.\
                                    loc[pre_trial_store['STORE_NBR']==trial_
                                        measuring_metric]).reset_index(drop=

        # Find correlation between two columns of data
        correlation_val = measuring_df.corrwith(df_forTrial, axis=0).values[

        # append the calculated values into a dataframe
        some_dataframe = some_dataframe.append({
            "trial_store": trial_store_num,
            "other_stores": i,
            "measure": measuring_metric,
            "correlation": correlation_val
        }, ignore_index=True)

    # return the dataframe created in this function
    return some_dataframe
```

In [133]:
```
# try store number 4
corr_nSales = findCorrelation(preTrialMeasures, 4, "totSales")
corr_nSales
```

Out[133]:

	trial_store	other_stores	measure	correlation
0	4	1	totSales	-0.513184
1	4	2	totSales	0.128407
2	4	3	totSales	0.227754
3	4	4	totSales	1.000000
4	4	5	totSales	-0.118735
...	...	...	...	...
255	4	268	totSales	-0.282850
256	4	269	totSales	0.463409
257	4	270	totSales	0.255360
258	4	271	totSales	0.318078
259	4	272	totSales	0.651821

260 rows × 4 columns

Apart from correlation, calculate a standardized metric based on the absolute difference bewteen the trial store's performance and each control store's performance.

In [131]:
```
# find magnitude difference
def findDifference(pre_trial_store, trial_store_num, measuring_metric):
    # create empty dataframe
    some_dataframe = pd.DataFrame(columns =
                                  ['trial_store',"other_stores","measure","di

    # define number of stores for looping
    number_of_stores = pre_trial_store['STORE_NBR'].unique()

    for i in number_of_stores:
        df_forTrial = pd.DataFrame(pre_trial_store.loc[pre_trial_store['STOR
        measuring_df = pd.DataFrame(pre_trial_store.\
                                    loc[pre_trial_store['STORE_NBR']==trial_
                                        measuring_metric]).reset_index(drop=
        # calculate absolute difference (this outputs a dataframe with total

        abs_difference = abs(measuring_df - df_forTrial)

        # standardize the magnitude distance so the measure ranges from 0 to
        ######################
        abs_max = abs_difference.max()
        abs_min = abs_difference.min()

        mag_difference = np.mean(1 - (abs_difference-abs_min) /
                                 (abs_max - abs_min))
        ######################

        # add calculated values into the dataframe
        some_dataframe = some_dataframe.append({
            "trial_store": trial_store_num,
            "other_stores": i,
            "measure": measuring_metric,
            "difference": mag_difference.values[0]
        }, ignore_index=True)


    return some_dataframe
```

In [134]:
```
# calculate correlation against store 77 using total sales and number of cus
```

In [135]:
```
preTrialMeasures.head()
```

Out[135]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	1.061224	1.192308	3.337097
1	1	201808	176.1	42	1.023810	1.255814	3.261111
2	1	201809	278.8	59	1.050847	1.209677	3.717333
3	1	201810	188.1	44	1.022727	1.288889	3.243103
4	1	201811	192.6	46	1.021739	1.212766	3.378947

In [937]:
```
# correlation for total sales and number of customers
corr_nSales = findCorrelation(preTrialMeasures, 77, "totSales")
corr_nCustomers = findCorrelation(preTrialMeasures, 77, "nCustomers")
```

In [938]:
```
corr_nSales
```

Out[938]:

	trial_store	other_stores	measure	correlation
0	77	1	totSales	0.075218
1	77	2	totSales	-0.263079
2	77	3	totSales	0.806644
3	77	4	totSales	-0.263300
4	77	5	totSales	-0.110652
...	...	...	...	...
255	77	268	totSales	0.344757
256	77	269	totSales	-0.315730
257	77	270	totSales	0.315430
258	77	271	totSales	0.355487
259	77	272	totSales	0.117622

260 rows × 4 columns

In [939]:
```
corr_nCustomers
```

Out[939]:

	trial_store	other_stores	measure	correlation
0	77	1	nCustomers	0.322168
1	77	2	nCustomers	-0.572051
2	77	3	nCustomers	0.834207
3	77	4	nCustomers	-0.295639
4	77	5	nCustomers	0.370659
...	...	...	...	...
255	77	268	nCustomers	0.369517
256	77	269	nCustomers	-0.474293
257	77	270	nCustomers	-0.131259
258	77	271	nCustomers	0.019629
259	77	272	nCustomers	0.223217

260 rows × 4 columns

In [940]:
```
# magnitude for total sales and number of customers
mag_nSales = findDifference(preTrialMeasures, 77, 'totSales')
mag_nCustomers = findDifference(preTrialMeasures, 77, "nCustomers")
```

In [941]:
```
mag_nSales
```

Out[941]:

	trial_store	other_stores	measure	difference
0	77	1	totSales	0.408163
1	77	2	totSales	0.590119
2	77	3	totSales	0.522914
3	77	4	totSales	0.644934
4	77	5	totSales	0.516320
...	...	...	...	...
255	77	268	totSales	0.429787
256	77	269	totSales	0.559099
257	77	270	totSales	0.591547
258	77	271	totSales	0.341091
259	77	272	totSales	0.523631

260 rows × 4 columns

In [942]:
```
mag_nCustomers
```

Out[942]:

	trial_store	other_stores	measure	difference
0	77	1	nCustomers	0.663866
1	77	2	nCustomers	0.471429
2	77	3	nCustomers	0.489796
3	77	4	nCustomers	0.498258
4	77	5	nCustomers	0.512605
...	...	...	...	...
255	77	268	nCustomers	0.571429
256	77	269	nCustomers	0.484472
257	77	270	nCustomers	0.536680
258	77	271	nCustomers	0.328571
259	77	272	nCustomers	0.545455

260 rows × 4 columns

In [943]:
```
# merge the dataframes with their respective measure metrics
corrMag_sales_df = pd.merge(corr_nSales, mag_nSales, how='inner')
corrMag_customers_df = pd.merge(corr_nCustomers, mag_nCustomers, how='inner'
corrMag_sales_df
```

Out[943]:

	trial_store	other_stores	measure	correlation	difference
0	77	1	totSales	0.075218	0.408163
1	77	2	totSales	-0.263079	0.590119
2	77	3	totSales	0.806644	0.522914
3	77	4	totSales	-0.263300	0.644934
4	77	5	totSales	-0.110652	0.516320
...	...	...	...	...	...
255	77	268	totSales	0.344757	0.429787
256	77	269	totSales	-0.315730	0.559099
257	77	270	totSales	0.315430	0.591547
258	77	271	totSales	0.355487	0.341091
259	77	272	totSales	0.117622	0.523631

260 rows × 5 columns

In [944]:
```
corrMag_customers_df
```

Out[944]:

	trial_store	other_stores	measure	correlation	difference
0	77	1	nCustomers	0.322168	0.663866
1	77	2	nCustomers	-0.572051	0.471429
2	77	3	nCustomers	0.834207	0.489796
3	77	4	nCustomers	-0.295639	0.498258
4	77	5	nCustomers	0.370659	0.512605
...	...	...	...	...	...
255	77	268	nCustomers	0.369517	0.571429
256	77	269	nCustomers	-0.474293	0.484472
257	77	270	nCustomers	-0.131259	0.536680
258	77	271	nCustomers	0.019629	0.328571
259	77	272	nCustomers	0.223217	0.545455

260 rows × 5 columns

In [945]:
```
# calculate weighted average of each row using correlation and difference
# use 50% weighting for now assuming the two variables are equally important
corr_weight = 0.5
# for the sales dataframe
corrMag_sales_df['corrMagAVG_sales'] = corr_weight * corrMag_sales_df['corre
                                       corr_weight * corrMag_sales_df['differen
# for the customers dataframe
corrMag_customers_df['corrMagAVG_customers'] = corr_weight * corrMag_custome
                                               corr_weight * corrMag_customers_df['
```

In [946]:
```
print("corrMag_sales_df")
display(corrMag_sales_df.head())
print("corrMag_customers_df")
corrMag_customers_df.head()
```

corrMag_sales_df

	trial_store	other_stores	measure	correlation	difference	corrMagAVG_sales
0	77	1	totSales	0.075218	0.408163	0.241691
1	77	2	totSales	-0.263079	0.590119	0.163520
2	77	3	totSales	0.806644	0.522914	0.664779
3	77	4	totSales	-0.263300	0.644934	0.190817
4	77	5	totSales	-0.110652	0.516320	0.202834

corrMag_customers_df

Out[946]:

	trial_store	other_stores	measure	correlation	difference	corrMagAVG_customers
0	77	1	nCustomers	0.322168	0.663866	0.493017
1	77	2	nCustomers	-0.572051	0.471429	-0.050311
2	77	3	nCustomers	0.834207	0.489796	0.662002
3	77	4	nCustomers	-0.295639	0.498258	0.101310
4	77	5	nCustomers	0.370659	0.512605	0.441632

In [947]:
```
corrMag_customers_df.loc[corrMag_customers_df['other_stores']==77]
```

Out[947]:

	trial_store	other_stores	measure	correlation	difference	corrMagAVG_customers
72	77	77	nCustomers	1.0	NaN	NaN

In [948]:
```
sum(corrMag_sales_df.index == corrMag_customers_df.index) == len(corrMag_sal
```

Out[948]: True

In [949]:
```
# with just the corrMagAVG column
s_df.reset_index(), corrMag_customers_df.reset_index(), how='inner')
oncat([corrMag_sales_df, corrMag_customers_df], axis=1)[['trial_store', 'other_st
```

Out[949]:

	trial_store	trial_store	other_stores	other_stores	corrMagAVG_sales	corrMagAVG_customers
0	77	77	1	1	0.241691	0.493017
1	77	77	2	2	0.163520	-0.050311
2	77	77	3	3	0.664779	0.662002
3	77	77	4	4	0.190817	0.101310
4	77	77	5	5	0.202834	0.441632

In [950]:
```
# remove duplicate columns
control_store_df = control_store_df.loc[:,~control_store_df.columns.duplicat
# check if changes are made
control_store_df
```

Out[950]:

	trial_store	other_stores	corrMagAVG_sales	corrMagAVG_customers
0	77	1	0.241691	0.493017
1	77	2	0.163520	-0.050311
2	77	3	0.664779	0.662002
3	77	4	0.190817	0.101310
4	77	5	0.202834	0.441632
...	...	...	...	...
255	77	268	0.387272	0.470473
256	77	269	0.121684	0.005090
257	77	270	0.453489	0.202710
258	77	271	0.348289	0.174100
259	77	272	0.320626	0.384336

260 rows × 4 columns

In [ ]:
```

```

store with the highest corrMagAVG score will be selected as the control store since it is the most similar to the trial store.

In [951]:
```
# define a column for the final control score
control_store_df['control_score'] = 0.5 * (control_store_df['corrMagAVG_sale
```

In [952]:
```
control_store_df.head()
```

Out[952]:

	trial_store	other_stores	corrMagAVG_sales	corrMagAVG_customers	control_score
0	77	1	0.241691	0.493017	0.367354
1	77	2	0.163520	-0.050311	0.056604
2	77	3	0.664779	0.662002	0.663390
3	77	4	0.190817	0.101310	0.146064
4	77	5	0.202834	0.441632	0.322233

In [953]:
```
# select control stores based on the highest matching store (close to 1 but
control_store_df.sort_values(by='control_score',ascending=False)

```

Out[953]:

	trial_store	other_stores	corrMagAVG_sales	corrMagAVG_customers	control_score
221	77	233	0.697290	0.816607	0.756949
67	77	71	0.789497	0.663123	0.726310
79	77	84	0.656972	0.715000	0.685986
111	77	119	0.636046	0.729729	0.682887
108	77	115	0.708347	0.645155	0.676751
...	...	...	...	...	...
230	77	242	-0.121818	-0.046675	-0.084247
178	77	186	-0.128806	-0.057268	-0.093037
95	77	102	-0.102238	-0.119121	-0.110680
8	77	9	-0.144121	-0.087135	-0.115628
72	77	77	NaN	NaN	NaN

260 rows × 5 columns

From this chart, we can see that store 233 matches trial store 77 the most.

In [954]:
```
# create a checkpoint for preTrialMeasures
store_data_df = preTrialMeasures.reset_index(drop=True)
store_data_df.head()
```

Out[954]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	1.061224	1.192308	3.337097
1	1	201808	176.1	42	1.023810	1.255814	3.261111
2	1	201809	278.8	59	1.050847	1.209677	3.717333
3	1	201810	188.1	44	1.022727	1.288889	3.243103
4	1	201811	192.6	46	1.021739	1.212766	3.378947

In [955]:
```
# set control store as store 233
ctrl_store = 233
# set trial store as 77
trial_store = 77
```

In [956]:
```
# add a column labelling whether the store is "other stores",'control store'
# create empty_list to store numbers
store_label = []
for i in range(store_data_df.shape[0]):
    if store_data_df['STORE_NBR'][i] == trial_store:
        store_label.append("trial_store")
    elif store_data_df['STORE_NBR'][i] == ctrl_store:
        store_label.append("control_store")
    else:
        store_label.append("other_stores")

store_data_df['store_type'] = store_label

```

In [957]:
```
store_data_df.head()
```

Out[957]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	
0	1	201807	206.9	49	1.061224	1.192308	3.337097	o
1	1	201808	176.1	42	1.023810	1.255814	3.261111	o
2	1	201809	278.8	59	1.050847	1.209677	3.717333	o
3	1	201810	188.1	44	1.022727	1.288889	3.243103	o
4	1	201811	192.6	46	1.021739	1.212766	3.378947	o

In [958]:
```
store_data_df['store_type'].unique()
```

Out[958]:
```
array(['other_stores', 'trial_store', 'control_store'], dtype=object)
```

In [959]:
```
# check if function was correct
store_data_df.loc[(store_data_df['store_type'] == "trial_store") | \
                  (store_data_df['store_type'] == "control_store")]
```

Out[959]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
504	77	201807	296.8	51	1.078431	1.527273	3.533333
505	77	201808	255.5	47	1.021277	1.541667	3.452703
506	77	201809	225.2	42	1.047619	1.590909	3.217143
507	77	201810	204.5	37	1.027027	1.368421	3.932692
508	77	201811	245.3	41	1.073171	1.522727	3.661194
509	77	201812	267.3	46	1.065217	1.469388	3.712500
510	77	201901	204.4	35	1.114286	1.666667	3.144615
1547	233	201807	290.7	51	1.058824	1.629630	3.303409
1548	233	201808	285.9	48	1.041667	1.600000	3.573750
1549	233	201809	228.6	42	1.071429	1.555556	3.265714
1550	233	201810	185.7	35	1.028571	1.555556	3.316071
1551	233	201811	211.6	40	1.025000	1.512195	3.412903
1552	233	201812	279.8	47	1.063830	1.500000	3.730667
1553	233	201901	177.5	35	1.000000	1.342857	3.776596

In [960]:
```
# run if needed
if "transaction_month" in store_data_df.columns.values:
    store_data_df.drop(columns='transaction_month', inplace=True)
```

In [961]:
```
# create a transaction month column for graphing later
date_data = pd.to_datetime(store_data_df['yearmonth'], format = '%Y%m')

# insert in the third index of the column beside yearmonth
store_data_df.insert(2, "transaction_month", date_data)
```

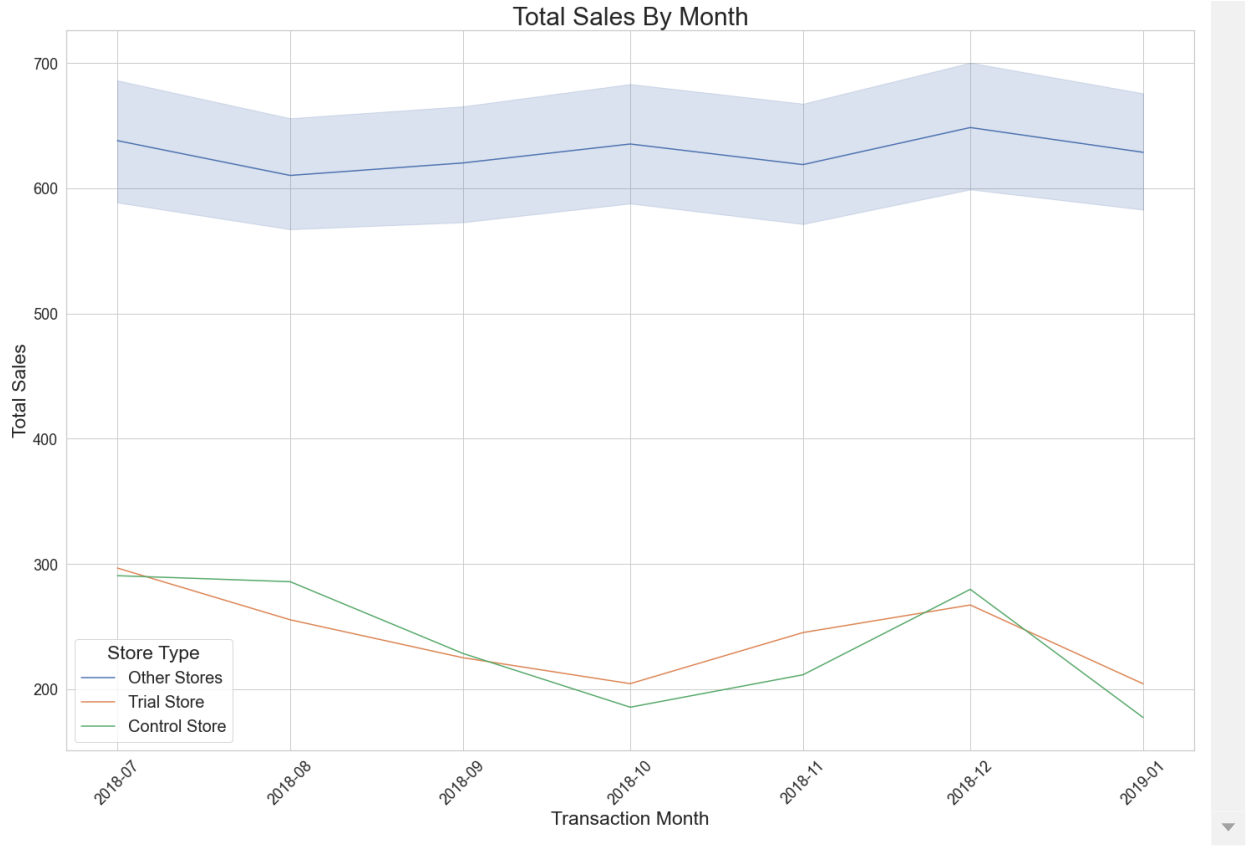
In [962]:
```
store_data_df.head()
```

Out[962]:

	STORE_NBR	yearmonth	transaction_month	totSales	nCustomers	nTxnPerCust	nChipsPerTxn
0	1	201807	2018-07-01	206.9	49	1.061224	1.192308
1	1	201808	2018-08-01	176.1	42	1.023810	1.255814
2	1	201809	2018-09-01	278.8	59	1.050847	1.209677
3	1	201810	2018-10-01	188.1	44	1.022727	1.288889
4	1	201811	2018-11-01	192.6	46	1.021739	1.212766

In [964]:

```
# visually check if drivers are indeed similar in the period before the tria
sns.set(style='whitegrid')
# set figure size
plt.figure(figsize=(20,14))

# create line chart
sns.lineplot(x = "transaction_month",
             y = "totSales",
             hue = "store_type",
             data = store_data_df)

# set x label orientation
plt.xticks(rotation=45)

# set fontsize
plt.xlabel("Transaction Month", fontsize=23)
plt.ylabel("Total Sales", fontsize=23)
plt.title("total sales by month".title(), fontsize=30)
plt.tick_params(labelsize=18)

# legend labels
some_labels = ['Other Stores',
               "Trial Store",
               "Control Store"]

# set legend location and fontsize
plt.legend(title = "store type".title(),
           labels=some_labels,
          loc='best',
          fontsize=20,
          title_fontsize=23)

# fit the graph
plt.tight_layout()

# save file
plt.savefig("static/module2_analysis_pics/trial_control_salesPerMonth.png")
```



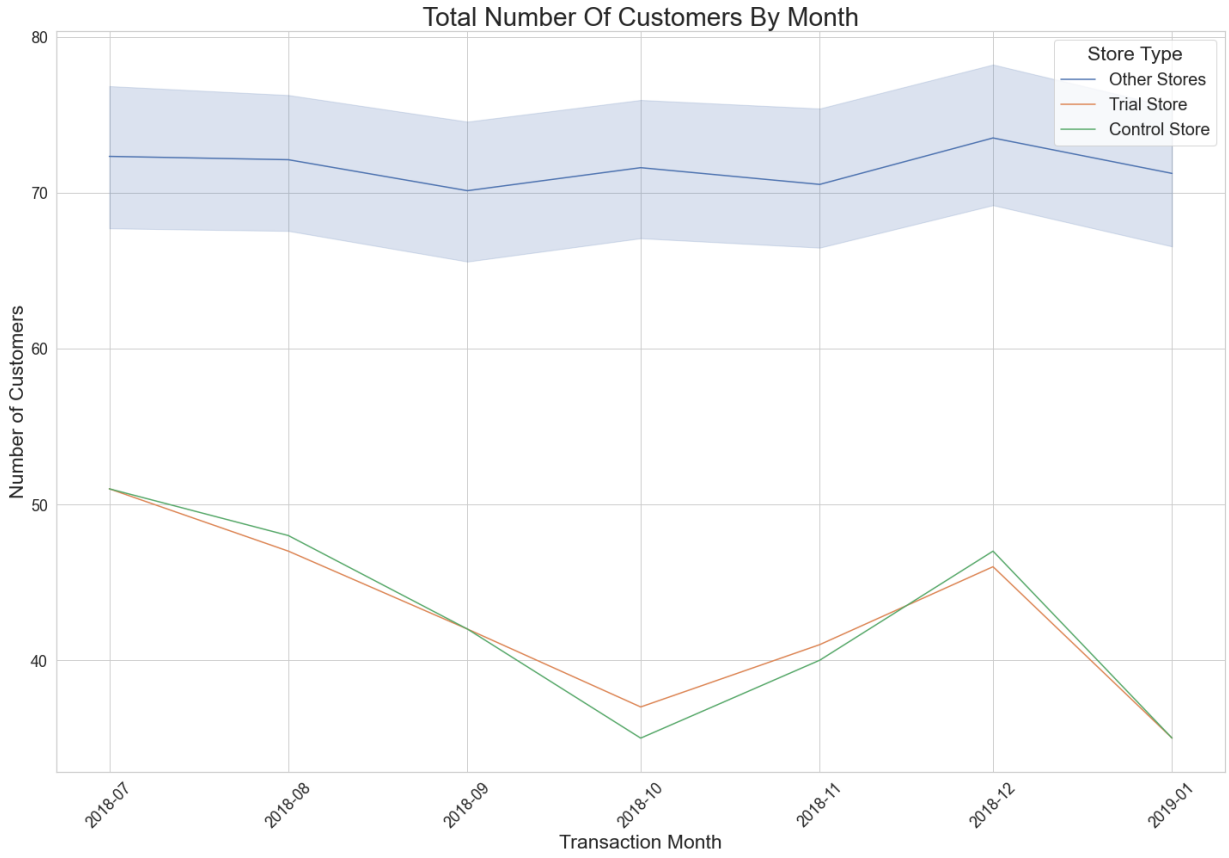
In [728]:
```
1 store_data_df.head(3)
```

Out[728]:

	STORE_NBR	yearmonth	transaction_month	totSales	nCustomers	nTxnPerCust	nChipsPerTxn
0	1	201807	2018-07-01	206.9	49	1.061224	1.192308
1	1	201808	2018-08-01	176.1	42	1.023810	1.255814
2	1	201809	2018-09-01	278.8	59	1.050847	1.209677

In [965]:

```
# this time graph the customers data
sns.set(style='whitegrid')
# set figure size
plt.figure(figsize=(20,14))

# create line chart
sns.lineplot(x = "transaction_month",
             y = "nCustomers",
             hue = "store_type",
             data = store_data_df)

# set x label orientation
plt.xticks(rotation=45)

# set fontsize
plt.xlabel("Transaction Month", fontsize=23)
plt.ylabel("Number of Customers", fontsize=23)
plt.title("total number of customers by month".title(), fontsize=30)
plt.tick_params(labelsize=18)

# legend labels
some_labels = ['Other Stores',
               "Trial Store",
               "Control Store"]

# set legend location and fontsize
plt.legend(title = "store type".title(),
           labels=some_labels,
          loc='best',
          fontsize=20,
          title_fontsize=23)

# fit the graph
plt.tight_layout()

# save file
plt.savefig("static/module2_analysis_pics/trial_control_customersPerMonth.pn

```



In [ ]:
```
1
```

Assement of Trial for trial store 77

The trial period starts from february 2019 to the end of April 2019 and now we check to see if there has been an uplift in overall chip sales

In [966]:
```
1 print(trial_store)
2 print(ctrl_store)
```

```
77
233
```

In [967]:
```
# scale pre_trial control sales to match pre-trial trial store sales

# trial store total sales
trial_totSales = store_data_df.loc[store_data_df['STORE_NBR'] == trial_store

# control store total sales
ctrl_totSales = store_data_df.loc[store_data_df['STORE_NBR'] == ctrl_store,

# scale numbers
scalingFactorForControlSales = trial_totSales / ctrl_totSales
scalingFactorForControlSales
```

Out[967]: 1.0236173032895528

In [968]:
```
measureOverTime_df.head()
```

Out[968]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	t
0	1	201807	206.9	49	1.061224	1.192308	3.337097	
1	1	201808	176.1	42	1.023810	1.255814	3.261111	
2	1	201809	278.8	59	1.050847	1.209677	3.717333	
3	1	201810	188.1	44	1.022727	1.288889	3.243103	
4	1	201811	192.6	46	1.021739	1.212766	3.378947	

In [969]:
```
# apply the scaling factor

scaled_control_stores = measureOverTime_df.loc[measureOverTime_df['STORE_NBR
scaled_control_stores
```

Out[969]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
2699	233	201807	290.7	51	1.058824	1.629630	3.303409
2700	233	201808	285.9	48	1.041667	1.600000	3.573750
2701	233	201809	228.6	42	1.071429	1.555556	3.265714
2702	233	201810	185.7	35	1.028571	1.555556	3.316071
2703	233	201811	211.6	40	1.025000	1.512195	3.412903
2704	233	201812	279.8	47	1.063830	1.500000	3.730667
2705	233	201901	177.5	35	1.000000	1.342857	3.776596
2706	233	201902	244.0	45	1.044444	1.489362	3.485714
2707	233	201903	199.1	40	1.025000	1.439024	3.374576
2708	233	201904	158.6	30	1.100000	1.393939	3.447826
2709	233	201905	344.4	57	1.087719	1.483871	3.743478
2710	233	201906	221.0	41	1.000000	1.487805	3.622951

In [970]:
```
# reorder columns
scaled_control_stores = scaled_control_stores[['STORE_NBR', 'yearmonth', 'nC
       'nChipsPerTxn', 'avgPricePerUnit', 'totSales']]

```

In [971]:
```
scaled_control_stores['control_sales'] = scaled_control_stores['totSales'] *
```

In [972]:
```
scaled_control_stores.head()
```

Out[972]:

	STORE_NBR	yearmonth	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	totSales
2699	233	201807	51	1.058824	1.629630	3.303409	290.7
2700	233	201808	48	1.041667	1.600000	3.573750	285.9
2701	233	201809	42	1.071429	1.555556	3.265714	228.6
2702	233	201810	35	1.028571	1.555556	3.316071	185.7
2703	233	201811	40	1.025000	1.512195	3.412903	211.6

In [973]:

```
## create a percentagedifference dataframe

# calcualte total sales for the trials data
trial_sales = measureOverTime_df.loc[measureOverTime_df['STORE_NBR'] == tria

# build the dataframe for analysis later
percentage_difference_df = scaled_control_stores[['yearmonth', 'control_sale
percentage_difference_df['trial_sales'] = trial_sales.values
percentage_difference_df
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

Out[973]:

	yearmonth	control_sales	trial_sales
2699	201807	297.565550	296.8
2700	201808	292.652187	255.5
2701	201809	233.998916	225.2
2702	201810	190.085733	204.5
2703	201811	216.597421	245.3
2704	201812	286.408121	267.3
2705	201901	181.692071	204.4
2706	201902	249.762622	235.0
2707	201903	203.802205	278.5
2708	201904	162.345704	263.5
2709	201905	352.533799	299.3
2710	201906	226.219424	264.7

In [974]:
```
## calculate percentage difference

# select control_sales
control2_sales = percentage_difference_df['control_sales']

# select trial sales
trial2_sales = percentage_difference_df['trial_sales']

# store calculated value in a new column
percentage_difference_df['percentage_diff'] = abs(control2_sales - trial2_sa

```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  # Remove the CWD from sys.path while we load stuff.
```

In [975]:
```
percentage_difference_df.reset_index(drop=True, inplace=True)
```

In [976]:
```
percentage_difference_df
```

Out[976]:

	yearmonth	control_sales	trial_sales	percentage_diff
0	201807	297.565550	296.8	0.002573
1	201808	292.652187	255.5	0.126950
2	201809	233.998916	225.2	0.037602
3	201810	190.085733	204.5	0.075830
4	201811	216.597421	245.3	0.132516
5	201812	286.408121	267.3	0.066716
6	201901	181.692071	204.4	0.124980
7	201902	249.762622	235.0	0.059107
8	201903	203.802205	278.5	0.366521
9	201904	162.345704	263.5	0.623080
10	201905	352.533799	299.3	0.151003
11	201906	226.219424	264.7	0.170103

the null hypothesis is that the trial period is the same as the pre-trial period, so i will verify this by checking to see if the difference is significant

In [977]:
```
from scipy.stats import ttest_ind
```

In [978]:
```
import scipy.stats
```

In [979]:
```
## take the standard deviation based on the scaled percentage difference in
perDiff_preTrial = percentage_difference_df.loc[percentage_difference_df['ye
stdDev = perDiff_preTrial.std()
stdDev
```

Out[979]: 0.04994076264142549

In [980]:
```
# test with null hypothesis of there being 0 difference between trial and co
percentage_difference_df['tValues'] = (percentage_difference_df['percentage_

```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

In [981]:
```
# select the trial period
percentage_difference_df.loc[percentage_difference_df['yearmonth'] >= 201902
```

Out[981]:

	yearmonth	control_sales	trial_sales	percentage_diff	tValues
7	201902	249.762622	235.0	0.059107	1.183534
8	201903	203.802205	278.5	0.366521	7.339116
9	201904	162.345704	263.5	0.623080	12.476373
10	201905	352.533799	299.3	0.151003	3.023650
11	201906	226.219424	264.7	0.170103	3.406093

In [982]:
```
# find the 95th percentile of the t distribution with degrees of freedom (df
# 95% confidence level
# degree of freedom of 7 is derived from have 9 pre-trial months. dof = 8-1=
scipy.stats.t.ppf(q=0.95, df=7)
```

Out[982]: 1.894578605061305

the t-value is much larger than the 95th percentile value of the t-distribution for March and April

The increase in sales in the trial store in March and April is statistically greater than in the control store.

Now we will create a more visual version of this by plotting the sales of the control store, the sales of the trial stores and the 95th percentile value of sales of the control store.

In [983]:
```
measureOverTime_df.head()
```

Out[983]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	t
0	1	201807	206.9	49	1.061224	1.192308	3.337097	
1	1	201808	176.1	42	1.023810	1.255814	3.261111	
2	1	201809	278.8	59	1.050847	1.209677	3.717333	
3	1	201810	188.1	44	1.022727	1.288889	3.243103	
4	1	201811	192.6	46	1.021739	1.212766	3.378947	

In [984]:
```
# create another checkpoint so we don't mess up the original data
measureOverTimeSales = measureOverTime_df
```

In [985]:
```
measureOverTimeSales.head()
```

Out[985]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	t
0	1	201807	206.9	49	1.061224	1.192308	3.337097	
1	1	201808	176.1	42	1.023810	1.255814	3.261111	
2	1	201809	278.8	59	1.050847	1.209677	3.717333	
3	1	201810	188.1	44	1.022727	1.288889	3.243103	
4	1	201811	192.6	46	1.021739	1.212766	3.378947	

In [986]:
```
# add a transaction month column
measureOverTimeSales['transaction_month'] = pd.to_datetime(measureOverTimeSa
```

In [987]:
```
# trial and control store total sales
trial_sales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'] ==
                                       ['transaction_month','totSales']].res

control_sales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'] =
                                         ['transaction_month','totSales']].re
```

In [988]:
```
display(trial_sales.head())
display(control_sales.head())
```

	transaction_month	totSales
0	2018-07-01	296.8
1	2018-08-01	255.5
2	2018-09-01	225.2
3	2018-10-01	204.5
4	2018-11-01	245.3

	transaction_month	totSales
0	2018-07-01	290.7
1	2018-08-01	285.9
2	2018-09-01	228.6
3	2018-10-01	185.7
4	2018-11-01	211.6

In [989]:
```
# rename columns
trial_sales.columns = ['transaction_month', 'trial_totSales']
control_sales.columns = ['transaction_month', 'control_totSales']
```

In [990]:
```
combineSales_df = pd.merge(trial_sales, control_sales, how='inner')
combineSales_df
```

Out[990]:

	transaction_month	trial_totSales	control_totSales
0	2018-07-01	296.8	290.7
1	2018-08-01	255.5	285.9
2	2018-09-01	225.2	228.6
3	2018-10-01	204.5	185.7
4	2018-11-01	245.3	211.6
5	2018-12-01	267.3	279.8
6	2019-01-01	204.4	177.5
7	2019-02-01	235.0	244.0
8	2019-03-01	278.5	199.1
9	2019-04-01	263.5	158.6
10	2019-05-01	299.3	344.4
11	2019-06-01	264.7	221.0

In [991]:
```
# create two columns marking the 95% and 5% confidence interval for graphing
combineSales_df['control_5%_interval'] = combineSales_df['control_totSales']

combineSales_df['control_95%_interval'] = combineSales_df['control_totSales'

combineSales_df
```

Out[991]:

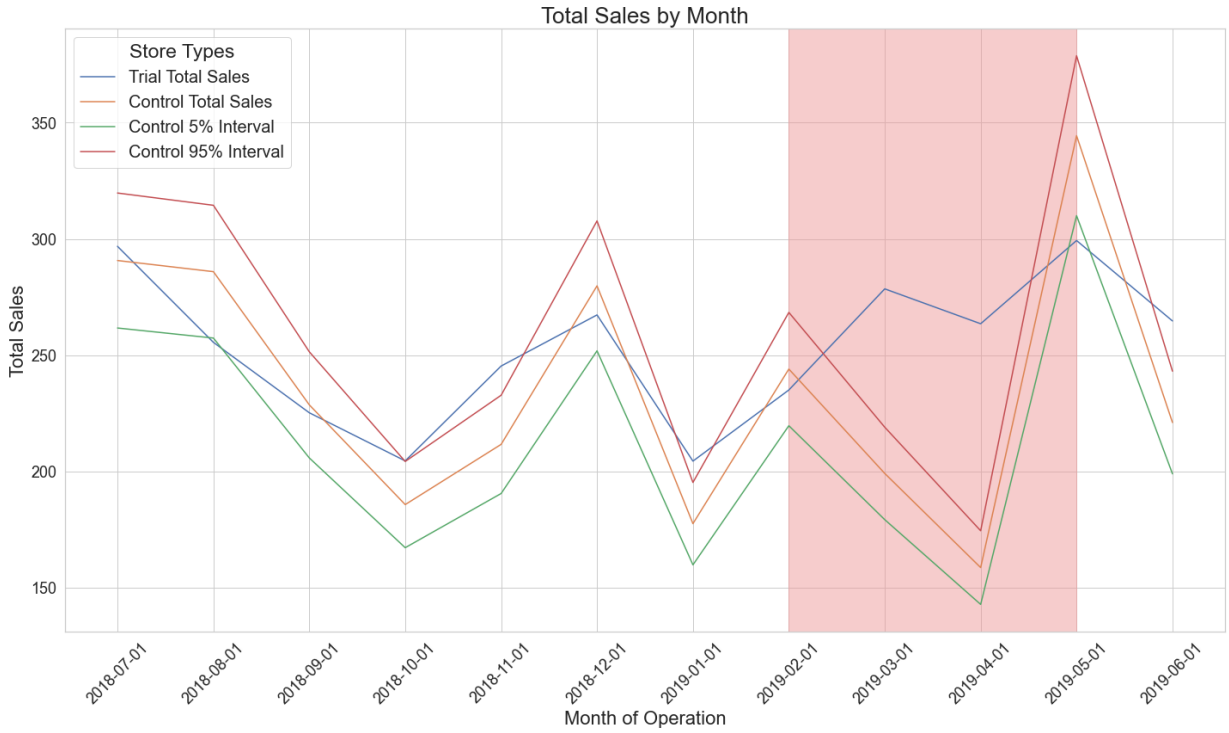
	transaction_month	trial_totSales	control_totSales	control_5%_interval	control_95%_interval
0	2018-07-01	296.8	290.7	261.664441	319.735559
1	2018-08-01	255.5	285.9	257.343872	314.456128
2	2018-09-01	225.2	228.6	205.767083	251.432917
3	2018-10-01	204.5	185.7	167.152001	204.247999
4	2018-11-01	245.3	211.6	190.465069	232.734931
5	2018-12-01	267.3	279.8	251.853149	307.746851
6	2019-01-01	204.4	177.5	159.771029	195.228971
7	2019-02-01	235.0	244.0	219.628908	268.371092
8	2019-03-01	278.5	199.1	179.213588	218.986412
9	2019-04-01	263.5	158.6	142.758790	174.441210
10	2019-05-01	299.3	344.4	310.000803	378.799197
11	2019-06-01	264.7	221.0	198.926183	243.073817

In [992]:
```
# melt combineSales_df so i can use the hue parameter later on seaborn
melted_77_sales_df = combineSales_df.melt('transaction_month', var_name='col

# convert transaction_month to string datatype so xticks won't display time
melted_77_sales_df['transaction_month'] = melted_77_sales_df['transaction_mo
```

In [993]:
```
# import dates
import matplotlib.dates as mdates
```

In [994]:

```
# plot
sns.set(style="whitegrid")

plt.figure(figsize=(20,12))

# plot
g = sns.lineplot(x="transaction_month",
                 y="vals",
                 hue='cols',
                 data= melted_77_sales_df
                 )

# Highlight trial period
g.axvspan("2019-02-01", "2019-05-01", color='#EF9A9A', alpha=0.5)

# set x labels orientation
plt.xticks(rotation=45)

# # set fontsize
plt.xlabel("Month of Operation", fontsize= 22)
plt.ylabel("Total Sales", fontsize=22)
plt.title("Total Sales by Month", fontsize=26)
plt.tick_params(labelsize=18)

# set x limits
# plt.xlim(["2018-07-01","2019-06-01"])

legend_labels = ['Trial Total Sales',
                 "Control Total Sales",
                 "Control 5% Interval",
                 "Control 95% Interval"]

plt.legend(title = "Store Types",
           labels = legend_labels,
          loc='best',
          fontsize=20,
          title_fontsize=23)

plt.tight_layout()

# save figure
plt.savefig("static/module2_analysis_pics/trail_77_salesByMonth.png")
```



From this graph we can conclude that the trial store 77 is significantly different than the control store in the trial period (2019-02-01 to 2019-06-01)

In [716]:
```
1 store_data_df.head()
```

Out[716]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	
0	1	201807	206.9	49	1.061224	1.192308	3.337097	o
1	1	201808	176.1	42	1.023810	1.255814	3.261111	o
2	1	201809	278.8	59	1.050847	1.209677	3.717333	o
3	1	201810	188.1	44	1.022727	1.288889	3.243103	o
4	1	201811	192.6	46	1.021739	1.212766	3.378947	o

In [734]:
```
1 print(trial_store)
2 print(ctrl_store)
```

```
77
233
```

In [745]:
```
# scale pre_trial control sales to match pre-trial trial store sales

# trial store total sales
trial_nCustomers = store_data_df.loc[store_data_df['STORE_NBR'] == trial_sto

# control store total sales
ctrl_nCustomers = store_data_df.loc[store_data_df['STORE_NBR'] == ctrl_store

# scale numbers
scalingFactorForControlSales = trial_nCustomers / ctrl_nCustomers
scalingFactorForControlSales
```

Out[745]: 1.0033557046979866

In [746]:
```
# apply the scaling factor
scaled_control_stores = measureOverTime_df.loc[measureOverTime_df['STORE_NBR
scaled_control_stores
```

Out[746]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
2699	233	201807	290.7	51	1.058824	1.629630	3.303409
2700	233	201808	285.9	48	1.041667	1.600000	3.573750
2701	233	201809	228.6	42	1.071429	1.555556	3.265714
2702	233	201810	185.7	35	1.028571	1.555556	3.316071
2703	233	201811	211.6	40	1.025000	1.512195	3.412903
2704	233	201812	279.8	47	1.063830	1.500000	3.730667
2705	233	201901	177.5	35	1.000000	1.342857	3.776596
2706	233	201902	244.0	45	1.044444	1.489362	3.485714
2707	233	201903	199.1	40	1.025000	1.439024	3.374576
2708	233	201904	158.6	30	1.100000	1.393939	3.447826
2709	233	201905	344.4	57	1.087719	1.483871	3.743478
2710	233	201906	221.0	41	1.000000	1.487805	3.622951

In [747]:
```
# reorder columns
scaled_control_stores = scaled_control_stores[['STORE_NBR', 'yearmonth', 'nC
       'nChipsPerTxn', 'avgPricePerUnit', 'totSales']]
```

In [751]:
```
# add a customers feature
scaled_control_stores['control_nCustomers'] = scaled_control_stores['nCustom
```

In [752]:
```
1 scaled_control_stores.head()
```

Out[752]:

	STORE_NBR	yearmonth	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	totSales
2699	233	201807	51	1.058824	1.629630	3.303409	290.7
2700	233	201808	48	1.041667	1.600000	3.573750	285.9
2701	233	201809	42	1.071429	1.555556	3.265714	228.6
2702	233	201810	35	1.028571	1.555556	3.316071	185.7
2703	233	201811	40	1.025000	1.512195	3.412903	211.6

In [754]:
```
1
2  ## create a percentagedifference dataframe
3
4  # calcualte total sales for the trials data
5  trial_nCustomers = measureOverTime_df.loc[measureOverTime_df['STORE_NBR'] ==
6
7  # build the dataframe for analysis later
8  percentage_difference_df = scaled_control_stores[['yearmonth', 'control_nCus
9  percentage_difference_df['trial_nCustomers'] = trial_nCustomers.values
10 percentage_difference_df
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

Out[754]:

	yearmonth	control_nCustomers	trial_nCustomers
2699	201807	51.171141	51
2700	201808	48.161074	47
2701	201809	42.140940	42
2702	201810	35.117450	37
2703	201811	40.134228	41
2704	201812	47.157718	46
2705	201901	35.117450	35
2706	201902	45.151007	45
2707	201903	40.134228	50
2708	201904	30.100671	47
2709	201905	57.191275	55
2710	201906	41.137584	41

In [755]:
```
## calculate percentage difference

# select control_sales
control2_nCustomers = percentage_difference_df['control_nCustomers']

# select trial sales
trial2_nCustomers = percentage_difference_df['trial_nCustomers']

# store calculated value in a new column
percentage_difference_df['percentage_diff'] = abs(control2_nCustomers - tria
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  # Remove the CWD from sys.path while we load stuff.
```

In [756]:
```
# reset index
percentage_difference_df.reset_index(drop=True, inplace=True)
```

In [757]:
```
percentage_difference_df
```

Out[757]:

	yearmonth	control_nCustomers	trial_nCustomers	percentage_diff
0	201807	51.171141	51	0.003344
1	201808	48.161074	47	0.024108
2	201809	42.140940	42	0.003344
3	201810	35.117450	37	0.053607
4	201811	40.134228	41	0.021572
5	201812	47.157718	46	0.024550
6	201901	35.117450	35	0.003344
7	201902	45.151007	45	0.003344
8	201903	40.134228	50	0.245819
9	201904	30.100671	47	0.561427
10	201905	57.191275	55	0.038315
11	201906	41.137584	41	0.003344

In [758]:
```
# Calculate standard deviation based on the scaled percentage difference in
perDiff_preTrial = percentage_difference_df.loc[percentage_difference_df['ye
stdDev = perDiff_preTrial.std()
stdDev
```

Out[758]: 0.01824074855824395

In [759]:
```
# test with null hypothesis of there being 0 difference between trial and co
percentage_difference_df['tValues'] = (percentage_difference_df['percentage_
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

In [760]:
```
percentage_difference_df.head()
```

Out[760]:

	yearmonth	control_nCustomers	trial_nCustomers	percentage_diff	tValues
0	201807	51.171141	51	0.003344	0.183352
1	201808	48.161074	47	0.024108	1.321664
2	201809	42.140940	42	0.003344	0.183352
3	201810	35.117450	37	0.053607	2.938874
4	201811	40.134228	41	0.021572	1.182622

In [ ]:
```

```

In [761]:
```
measureOverTimeSales.head()
```

Out[761]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	t
0	1	201807	206.9	49	1.061224	1.192308	3.337097	
1	1	201808	176.1	42	1.023810	1.255814	3.261111	
2	1	201809	278.8	59	1.050847	1.209677	3.717333	
3	1	201810	188.1	44	1.022727	1.288889	3.243103	
4	1	201811	192.6	46	1.021739	1.212766	3.378947	

In [762]:
```
# trial and control store number of customers
trial_nCustomers = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'
                                            ['transaction_month','nCustomers']].r

control_nCustomers = measureOverTimeSales.loc[measureOverTimeSales['STORE_NB
                                              ['transaction_month','nCustomers']].
```

In [763]:
```
display(trial_nCustomers)
display(control_nCustomers.head())
```

	transaction_month	nCustomers
0	2018-07-01	51
1	2018-08-01	47
2	2018-09-01	42
3	2018-10-01	37
4	2018-11-01	41
5	2018-12-01	46
6	2019-01-01	35
7	2019-02-01	45
8	2019-03-01	50
9	2019-04-01	47
10	2019-05-01	55
11	2019-06-01	41

	transaction_month	nCustomers
0	2018-07-01	51
1	2018-08-01	48
2	2018-09-01	42
3	2018-10-01	35
4	2018-11-01	40

In [764]:
```
# rename columns
trial_nCustomers.columns = ['transaction_month', 'trial_nCustomers']
control_nCustomers.columns = ['transaction_month', 'control_nCustomers']
```

In [765]:
```
# merge the two dataframes
combineCustomers_df = pd.merge(trial_nCustomers, control_nCustomers, how='in
combineCustomers_df.head()
```

Out[765]:

	transaction_month	trial_nCustomers	control_nCustomers
0	2018-07-01	51	51
1	2018-08-01	47	48
2	2018-09-01	42	42
3	2018-10-01	37	35
4	2018-11-01	41	40

In [766]:
```
# create two columns marking the confidence interval
combineCustomers_df['control_5%_interval'] = combineCustomers_df['control_nC
combineCustomers_df['control_95%_interval'] = combineCustomers_df['control_n

```

In [767]:
```
combineCustomers_df.head()
```

Out[767]:

	transaction_month	trial_nCustomers	control_nCustomers	control_5%_interval	control_95%_inter
0	2018-07-01	51	51	49.139444	52.8605
1	2018-08-01	47	48	46.248888	49.7511
2	2018-09-01	42	42	40.467777	43.5322
3	2018-10-01	37	35	33.723148	36.2768
4	2018-11-01	41	40	38.540740	41.4592

In [768]:
```
# melt combineCustomers_df for graphing later
melted_77_customer_df = combineCustomers_df.melt('transaction_month', var_na

# convert transaction month to string datatype so xticks won't display time
melted_77_customer_df['transaction_month'] = melted_77_customer_df['transact
```

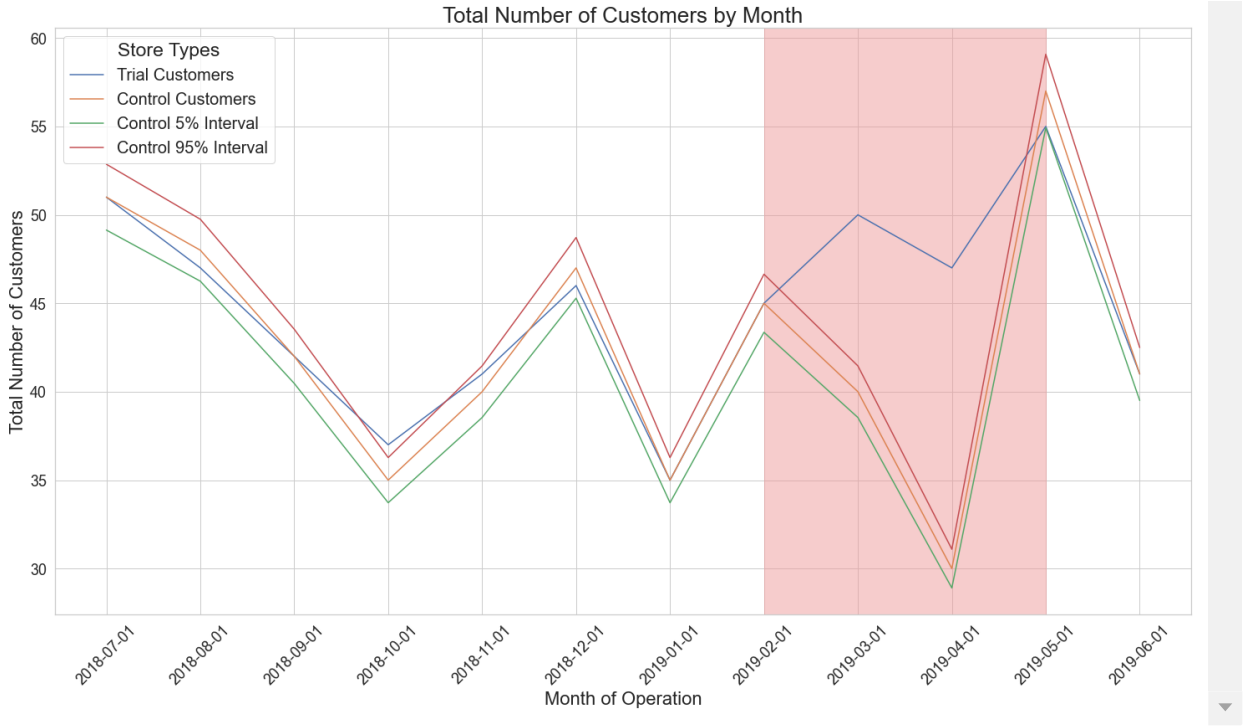
In [769]:
```
# look at the first and last 5 datapoints
display(melted_77_customer_df.head())
display(melted_77_customer_df.tail())
```

	transaction_month	cols	vals
0	2018-07-01	trial_nCustomers	51.0
1	2018-08-01	trial_nCustomers	47.0
2	2018-09-01	trial_nCustomers	42.0
3	2018-10-01	trial_nCustomers	37.0
4	2018-11-01	trial_nCustomers	41.0

	transaction_month	cols	vals
43	2019-02-01	control_95%_interval	46.641667
44	2019-03-01	control_95%_interval	41.459260
45	2019-04-01	control_95%_interval	31.094445
46	2019-05-01	control_95%_interval	59.079445
47	2019-06-01	control_95%_interval	42.495741

In [995]:

```
# plot
sns.set(style='whitegrid')

# set figure size
plt.figure(figsize=(20,12))

# plot
ax = sns.lineplot(x='transaction_month',
                  y = 'vals',
                  hue = 'cols',
                  data = melted_77_customer_df)

# highlight trial period
ax.axvspan("2019-02-01",'2019-05-01',color = '#EF9A9A', alpha=0.5)

# set x labels orientation for visability
plt.xticks(rotation=45)

# set font sizes and labels
plt.xlabel("Month of Operation", fontsize = 22)
plt.ylabel("Total Number of Customers", fontsize = 22)
plt.title("Total Number of Customers by Month", fontsize=26)
plt.tick_params(labelsize=18)

legend_labels = ['Trial Customers',
                 "Control Customers",
                 "Control 5% Interval",
                 "Control 95% Interval"]

plt.legend(title = "Store Types",
           labels = legend_labels,
          loc='best',
          fontsize=20,
          title_fontsize=23)

plt.tight_layout()
plt.savefig("static/module2_analysis_pics/trial_77_customerByMonth.png")
```



The number of customers for the trial store (store 77) is much higher than the control store during the trial period similar to the sales graph we identified earlier.

Other trial stores we have are store 86 and store 88 so i will now assess the impact for each of these two stores

Trial Store 86

In [997]:
```
trial_store = 86
# find correlation for sales and customers using previously defined function
corr_nSales = findCorrelation(preTrialMeasures, trial_store, "totSales")
corr_nCustomers = findCorrelation(preTrialMeasures, trial_store, 'nCustomers
```

In [998]:
```
# compute magnitude with new trial store
magnitude_nSales = findDifference(preTrialMeasures, trial_store, 'totSales')
magnitude_nCustomers = findDifference(preTrialMeasures, trial_store, 'nCusto
```

In [999]:
```
corr_nSales
```

Out[999]:

	trial_store	other_stores	measure	correlation
0	86	1	totSales	0.445632
1	86	2	totSales	-0.403835
2	86	3	totSales	-0.261284
3	86	4	totSales	-0.039035
4	86	5	totSales	0.235159
...	...	...	...	...
255	86	268	totSales	-0.452182
256	86	269	totSales	0.697055
257	86	270	totSales	-0.730679
258	86	271	totSales	0.527637
259	86	272	totSales	0.004926

260 rows × 4 columns

In [1000]:
```
# concatenate the sales together and customers together
corrMag_sales_df = pd.merge(corr_nSales, magnitude_nSales, how='inner')
corrMag_customers_df = pd.merge(corr_nCustomers, magnitude_nCustomers, how='
corrMag_sales_df.head()
```

Out[1000]:

	trial_store	other_stores	measure	correlation	difference
0	86	1	totSales	0.445632	0.488334
1	86	2	totSales	-0.403835	0.321131
2	86	3	totSales	-0.261284	0.507515
3	86	4	totSales	-0.039035	0.635654
4	86	5	totSales	0.235159	0.579835

In [1001]:
```
# calculate weighted average of each row using correlation and difference
# use 50% weighting for now assuming the two variables are equally important
corr_weight = 0.5
# for the sales dataframe
corrMag_sales_df['corrMagAVG_sales'] = corr_weight * corrMag_sales_df['corre
                                       corr_weight * corrMag_sales_df['differen
# for the customers dataframe
corrMag_customers_df['corrMagAVG_customers'] = corr_weight * corrMag_custome
                                               corr_weight * corrMag_customers_df['
```

In [1002]:
```
print("corrMag_sales_df")
display(corrMag_sales_df.head())
print("corrMag_customers_df")
corrMag_customers_df.head()
```

corrMag_sales_df

	trial_store	other_stores	measure	correlation	difference	corrMagAVG_sales
0	86	1	totSales	0.445632	0.488334	0.466983
1	86	2	totSales	-0.403835	0.321131	-0.041352
2	86	3	totSales	-0.261284	0.507515	0.123116
3	86	4	totSales	-0.039035	0.635654	0.298309
4	86	5	totSales	0.235159	0.579835	0.407497

corrMag_customers_df

Out[1002]:

	trial_store	other_stores	measure	correlation	difference	corrMagAVG_customers
0	86	1	nCustomers	0.485831	0.510204	0.498018
1	86	2	nCustomers	-0.086161	0.428571	0.171205
2	86	3	nCustomers	-0.353786	0.563025	0.104620
3	86	4	nCustomers	-0.169608	0.537815	0.184103
4	86	5	nCustomers	-0.253229	0.714286	0.230528

In [1003]:
```
### FIND CONTROL STORE ###

# merge into one dataframe with just the corrMagAVG column
# pd.merge(corrMag_sales_df.reset_index(), corrMag_customers_df.reset_index(
control_store_df = pd.concat([corrMag_sales_df, corrMag_customers_df], axis=
control_store_df.head()
```

Out[1003]:

	trial_store	trial_store	other_stores	other_stores	corrMagAVG_sales	corrMagAVG_customers
0	86	86	1	1	0.466983	0.498018
1	86	86	2	2	-0.041352	0.171205
2	86	86	3	3	0.123116	0.104620
3	86	86	4	4	0.298309	0.184103
4	86	86	5	5	0.407497	0.230528

In [1004]:
```
# remove duplicate columns
control_store_df = control_store_df.loc[:, ~control_store_df.columns.duplica
```

In [1005]:
```
# check if changes are made
control_store_df.head()
```

Out[1005]:

	trial_store	other_stores	corrMagAVG_sales	corrMagAVG_customers
0	86	1	0.466983	0.498018
1	86	2	-0.041352	0.171205
2	86	3	0.123116	0.104620
3	86	4	0.298309	0.184103
4	86	5	0.407497	0.230528

In [1006]:
```
# Store with the highest corrMagAVG score will be selected as the control st
control_store_df['control_store'] = 0.5 * (control_store_df['corrMagAVG_sale
                                           control_store_df['corrMagAVG_cust
```

In [1007]:
```
control_store_df.head()
```

Out[1007]:

	trial_store	other_stores	corrMagAVG_sales	corrMagAVG_customers	control_store
0	86	1	0.466983	0.498018	0.482500
1	86	2	-0.041352	0.171205	0.064927
2	86	3	0.123116	0.104620	0.113868
3	86	4	0.298309	0.184103	0.241206
4	86	5	0.407497	0.230528	0.319013

In [1008]:
```
# select the control store based on the highest score
control_store_df.sort_values(by='control_store', ascending=False).head()
```

Out[1008]:

	trial_store	other_stores	corrMagAVG_sales	corrMagAVG_customers	control_store
147	86	155	0.808106	0.733343	0.770724
102	86	109	0.697120	0.742532	0.719826
107	86	114	0.631393	0.663384	0.647389
213	86	225	0.601841	0.684356	0.643099
130	86	138	0.593296	0.660565	0.626930

From this chart, since store 155 has the highest score, we will select store 155 as our control store.

In [1009]:
```
# create checkpoint for preTrialMeasures
store_data_df = preTrialMeasures.reset_index(drop=True)
store_data_df.head()
```

Out[1009]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	1.061224	1.192308	3.337097
1	1	201808	176.1	42	1.023810	1.255814	3.261111
2	1	201809	278.8	59	1.050847	1.209677	3.717333
3	1	201810	188.1	44	1.022727	1.288889	3.243103
4	1	201811	192.6	46	1.021739	1.212766	3.378947

In [1010]:
```
# set trial and control stores
ctrl_store = 155
print(ctrl_store)
trial_store
```

```
155
```

Out[1010]:
```
86
```

In [1011]:
```
# select just the trial and control store
store_data_df.loc[(store_data_df['STORE_NBR'] == ctrl_store) |
                  (store_data_df['STORE_NBR'] == trial_store)]
```

Out[1011]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
560	86	201807	892.20	99	1.272727	1.992063	3.554582
561	86	201808	764.05	94	1.191489	1.919643	3.553721
562	86	201809	914.60	103	1.252427	2.000000	3.544961
563	86	201810	948.40	109	1.266055	2.000000	3.436232
564	86	201811	918.00	100	1.270000	2.000000	3.614173
565	86	201812	841.20	98	1.224490	2.000000	3.505000
566	86	201901	841.40	94	1.382979	2.000000	3.236154
1029	155	201807	924.60	101	1.237624	2.000000	3.698400
1030	155	201808	782.70	91	1.318681	1.908333	3.417904
1031	155	201809	1014.40	103	1.407767	2.000000	3.497931
1032	155	201810	963.80	108	1.259259	2.000000	3.543382
1033	155	201811	898.80	101	1.336634	2.000000	3.328889
1034	155	201812	849.80	97	1.247423	2.000000	3.511570
1035	155	201901	874.60	96	1.312500	2.000000	3.470635

In [1012]:
```
# create store types for graphing later (store type as in trial/control/othe
store_label = []
for i in range(store_data_df.shape[0]):
    if store_data_df['STORE_NBR'][i] == trial_store:
        store_label.append("trial_store")
    elif store_data_df['STORE_NBR'][i] == ctrl_store:
        store_label.append("control_store")
    else:
        store_label.append("other_stores")

store_data_df['store_type'] = store_label
```

In [1013]:
```
# add a date column to store_data_df
store_data_df['transaction_dates'] = pd.to_datetime(store_data_df['yearmonth
```

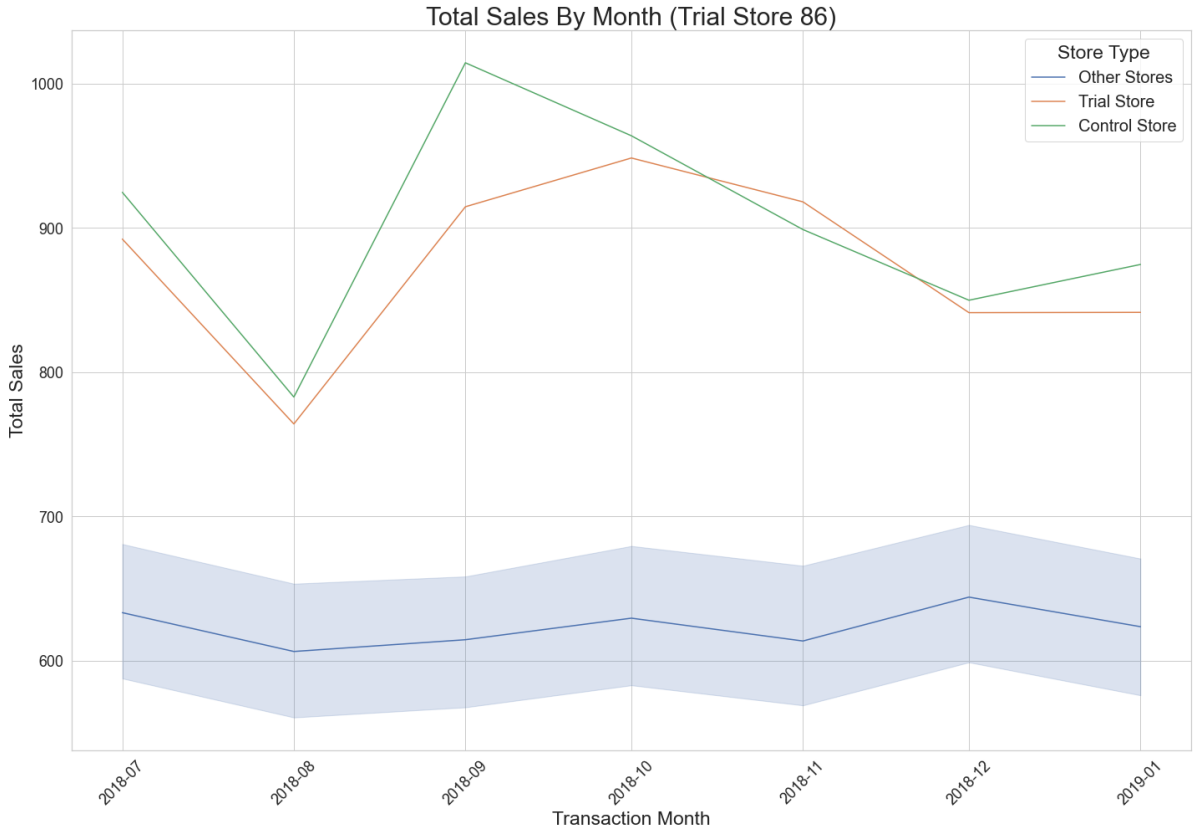
In [1014]:
```
store_data_df.head()
```

Out[1014]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	
0	1	201807	206.9	49	1.061224	1.192308	3.337097	o
1	1	201808	176.1	42	1.023810	1.255814	3.261111	o
2	1	201809	278.8	59	1.050847	1.209677	3.717333	o
3	1	201810	188.1	44	1.022727	1.288889	3.243103	o
4	1	201811	192.6	46	1.021739	1.212766	3.378947	o

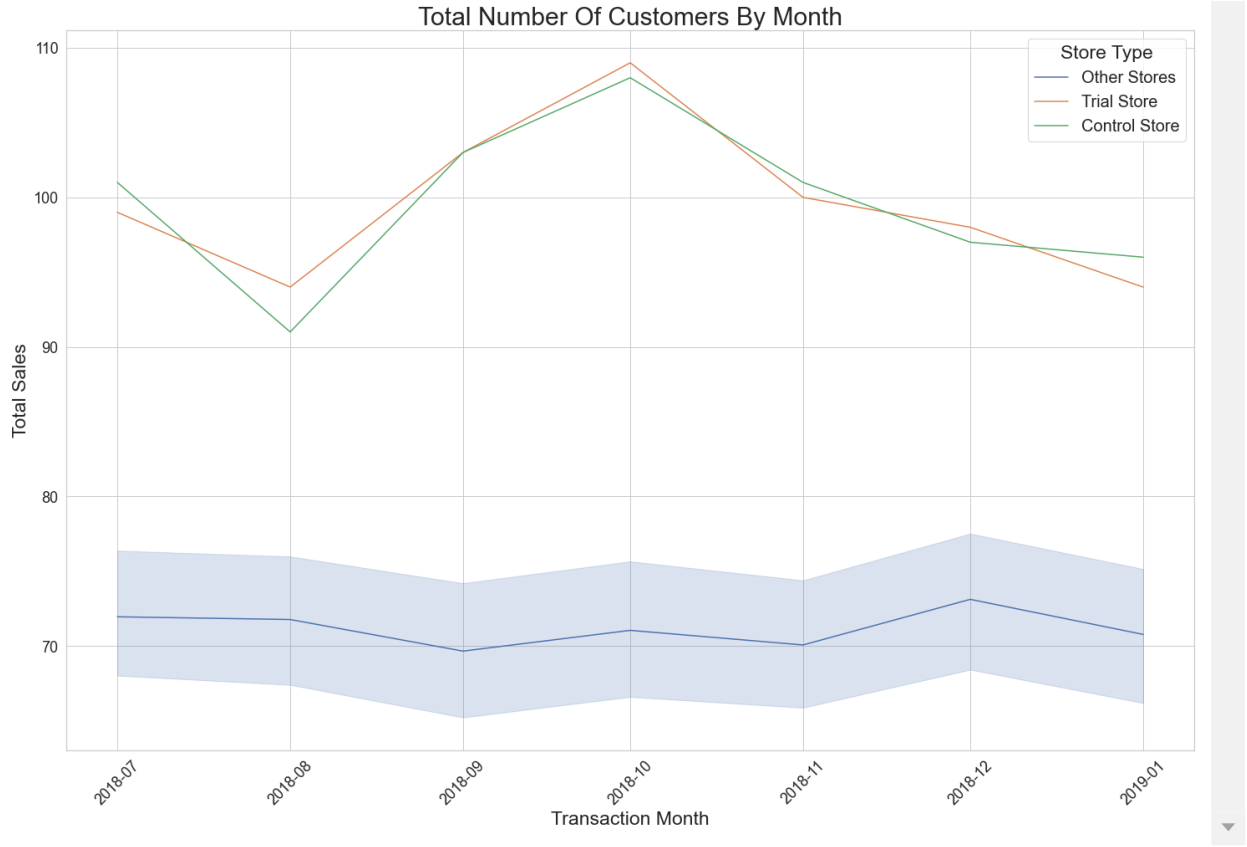
In [1015]:

```
# visually check if drivers are indeed similar in the period before the tria
sns.set(style='whitegrid')
# set figure size
plt.figure(figsize=(20,14))

# create line chart
sns.lineplot(x = "transaction_dates",
             y = "totSales",
             hue = "store_type",
             data = store_data_df)

# set x label orientation
plt.xticks(rotation=45)

# set fontsize
plt.xlabel("Transaction Month", fontsize=23)
plt.ylabel("Total Sales", fontsize=23)
plt.title("total sales by month (trial store 86)".title(), fontsize=30)
plt.tick_params(labelsize=18)

# legend labels
some_labels = ['Other Stores',
               "Trial Store",
               "Control Store"]

# set legend location and fontsize
plt.legend(title = "store type".title(),
           labels=some_labels,
          loc='best',
          fontsize=20,
          title_fontsize=23)

# fit the graph
plt.tight_layout()

# save figure
plt.savefig("static/module2_analysis_pics/trial_control_86_salesPerMonth.png
```


Total Sales By Month (Trial Store 86)
Store Type
Other Stores
Trial Store
Control Store
Total Sales
1000
900
800
700
600
2018-07
2018-08
2018-09
2018-10
2018-11
2018-12
2019-01
Transaction Month

In [1016]:

```
# graph the customers data
# visually check if drivers are indeed similar in the period before the tria
sns.set(style='whitegrid')
# set figure size
plt.figure(figsize=(20,14))

# create line chart
sns.lineplot(x = "transaction_dates",
             y = "nCustomers",
             hue = "store_type",
             data = store_data_df)

# set x label orientation
plt.xticks(rotation=45)

# set fontsize
plt.xlabel("Transaction Month", fontsize=23)
plt.ylabel("Total Sales", fontsize=23)
plt.title("total Number of Customers by month".title(), fontsize=30)
plt.tick_params(labelsize=18)

# legend labels
some_labels = ['Other Stores',
               "Trial Store",
               "Control Store"]

# set legend location and fontsize
plt.legend(title = "store type".title(),
           labels=some_labels,
          loc='best',
          fontsize=20,
          title_fontsize=23)

# fit the graph
plt.tight_layout()

#save figure
plt.savefig("static/module2_analysis_pics/trial_control_86_customerPerMonth.
```



Assessment for trial store 86

In [806]:
```
1 trial_store
```

Out[806]: 86

In [807]:
```
1  # scale pre_trial control sales to match pre-trial store sales
2
3  # trial store total sales
4  trial_totSales = store_data_df.loc[store_data_df['STORE_NBR'] == trial_store
5
6  # control store total sales
7  ctrl_totSales = store_data_df.loc[store_data_df['STORE_NBR'] == ctrl_store,
8
9  # scale numbers
10 scalingFactorForControlSales = trial_totSales / ctrl_totSales
11 scalingFactorForControlSales
```

Out[807]: 0.9700651481287746

In [808]:
```
# apply the scaling factor

scaled_control_stores = measureOverTime_df.loc[measureOverTime_df['STORE_NBR
scaled_control_stores
```

Out[808]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
1793	155	201807	924.60	101	1.237624	2.000000	3.698400
1794	155	201808	782.70	91	1.318681	1.908333	3.417904
1795	155	201809	1014.40	103	1.407767	2.000000	3.497931
1796	155	201810	963.80	108	1.259259	2.000000	3.543382
1797	155	201811	898.80	101	1.336634	2.000000	3.328889
1798	155	201812	849.80	97	1.247423	2.000000	3.511570
1799	155	201901	874.60	96	1.312500	2.000000	3.470635
1800	155	201902	891.20	95	1.336842	2.000000	3.508661
1801	155	201903	804.40	94	1.276596	2.000000	3.351667
1802	155	201904	844.60	99	1.222222	2.000000	3.490083
1803	155	201905	922.85	106	1.292453	1.934307	3.482453
1804	155	201906	857.20	95	1.284211	2.000000	3.513115

In [809]:
```
# reorder columns
scaled_control_stores = scaled_control_stores[['STORE_NBR', 'yearmonth', 'nC
       'nChipsPerTxn', 'avgPricePerUnit', 'totSales']]
```

In [810]:
```
scaled_control_stores['control_sales'] = scaled_control_stores['totSales'] *
```

In [811]:
```
1 scaled_control_stores
```

Out[811]:

	STORE_NBR	yearmonth	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	totSales
1793	155	201807	101	1.237624	2.000000	3.698400	924.60
1794	155	201808	91	1.318681	1.908333	3.417904	782.70
1795	155	201809	103	1.407767	2.000000	3.497931	1014.40
1796	155	201810	108	1.259259	2.000000	3.543382	963.80
1797	155	201811	101	1.336634	2.000000	3.328889	898.80
1798	155	201812	97	1.247423	2.000000	3.511570	849.80
1799	155	201901	96	1.312500	2.000000	3.470635	874.60
1800	155	201902	95	1.336842	2.000000	3.508661	891.20
1801	155	201903	94	1.276596	2.000000	3.351667	804.40
1802	155	201904	99	1.222222	2.000000	3.490083	844.60
1803	155	201905	106	1.292453	1.934307	3.482453	922.85
1804	155	201906	95	1.284211	2.000000	3.513115	857.20

In [812]:

```
## create a percentagedifference dataframe

# calcualte total sales for the trials data
trial_sales = measureOverTime_df.loc[measureOverTime_df['STORE_NBR'] == tria

# build the dataframe for analysis later
percentage_difference_df = scaled_control_stores[['yearmonth', 'control_sale
percentage_difference_df['trial_sales'] = trial_sales.values
percentage_difference_df
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

Out[812]:

	yearmonth	control_sales	trial_sales
1793	201807	896.922236	892.20
1794	201808	759.269991	764.05
1795	201809	984.034086	914.60
1796	201810	934.948790	948.40
1797	201811	871.894555	918.00
1798	201812	824.361363	841.20
1799	201901	848.418979	841.40
1800	201902	864.522060	913.20
1801	201903	780.320405	1026.80
1802	201904	819.317024	848.20
1803	201905	895.224622	889.30
1804	201906	831.539845	838.00

In [813]:
```
## calculate percentage difference

# select control_sales
control2_sales = percentage_difference_df['control_sales']

# select trial sales
trial2_sales = percentage_difference_df['trial_sales']

# store calculated value in a new column
percentage_difference_df['percentage_diff'] = abs(control2_sales - trial2_sa
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  # Remove the CWD from sys.path while we load stuff.
```

In [814]:
```
percentage_difference_df.reset_index(drop=True, inplace=True)
```

In [815]:
```
percentage_difference_df
```

Out[815]:

	yearmonth	control_sales	trial_sales	percentage_diff
0	201807	896.922236	892.20	0.005265
1	201808	759.269991	764.05	0.006296
2	201809	984.034086	914.60	0.070561
3	201810	934.948790	948.40	0.014387
4	201811	871.894555	918.00	0.052880
5	201812	824.361363	841.20	0.020426
6	201901	848.418979	841.40	0.008273
7	201902	864.522060	913.20	0.056306
8	201903	780.320405	1026.80	0.315870
9	201904	819.317024	848.20	0.035253
10	201905	895.224622	889.30	0.006618
11	201906	831.539845	838.00	0.007769

Our null hypothesis is such that the trial period is the same as the pre-trial period Take the standard deviation based on the scaled percentage difference in the pre-trial period

In [816]:
```
perDiff_preTrial = percentage_difference_df.loc[percentage_difference_df['ye
stdDev = perDiff_preTrial.std()
stdDev
```

Out[816]: 0.025833952854772656

In [817]:
```
# test with null hypothesis of there being 0 difference between trial and co
percentage_difference_df['tValues'] = (percentage_difference_df['percentage_
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

In [818]:
```
# select the trial period
percentage_difference_df.loc[percentage_difference_df['yearmonth'] >= 201902
```

Out[818]:

	yearmonth	control_sales	trial_sales	percentage_diff	tValues
7	201902	864.522060	913.2	0.056306	2.179542
8	201903	780.320405	1026.8	0.315870	12.226922
9	201904	819.317024	848.2	0.035253	1.364580
10	201905	895.224622	889.3	0.006618	0.256176
11	201906	831.539845	838.0	0.007769	0.300725

In [819]:
```
# find the 95th percentile of the t distribution with degrees of freedom (df
# 95% confidence level
# degree of freedom of 7 is derived from have 9 pre-trial months. dof = 8-1=
scipy.stats.t.ppf(q=0.95, df=7)
```

Out[819]: 1.894578605061305

In [820]:
```
# create another checkpoint so we don't mess up the original data
measureOverTimeSales = measureOverTime_df
```

In [821]:
```

# add a transaction month column
measureOverTimeSales['transaction_month'] = pd.to_datetime(measureOverTimeSa
```

In [822]:
```
# trial and control store total sales
trial_sales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'] ==
                                       ['transaction_month','totSales']].res

control_sales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'] =
                                        ['transaction_month','totSales']].re
```

In [823]:
```
display(trial_sales.head())
display(control_sales.head())
```

	transaction_month	totSales
0	2018-07-01	892.20
1	2018-08-01	764.05
2	2018-09-01	914.60
3	2018-10-01	948.40
4	2018-11-01	918.00

	transaction_month	totSales
0	2018-07-01	924.6
1	2018-08-01	782.7
2	2018-09-01	1014.4
3	2018-10-01	963.8
4	2018-11-01	898.8

In [824]:
```
# rename columns for trial and control data
trial_sales.columns = ['transaction_month', 'trial_totSales']
control_sales.columns = ['transaction_month', 'control_totSales']
```

In [825]:
```
combineSales_df = pd.merge(trial_sales, control_sales, how='inner')
combineSales_df
```

Out[825]:

	transaction_month	trial_totSales	control_totSales
0	2018-07-01	892.20	924.60
1	2018-08-01	764.05	782.70
2	2018-09-01	914.60	1014.40
3	2018-10-01	948.40	963.80
4	2018-11-01	918.00	898.80
5	2018-12-01	841.20	849.80
6	2019-01-01	841.40	874.60
7	2019-02-01	913.20	891.20
8	2019-03-01	1026.80	804.40
9	2019-04-01	848.20	844.60
10	2019-05-01	889.30	922.85
11	2019-06-01	838.00	857.20

In [826]:
```
# create two columns marking the 95% and 5% confidence interval for graphing
combineSales_df['control_5%_interval'] = combineSales_df['control_totSales']

combineSales_df['control_95%_interval'] = combineSales_df['control_totSales'

combineSales_df
```

Out[826]:

	transaction_month	trial_totSales	control_totSales	control_5%_interval	control_95%_interval
0	2018-07-01	892.20	924.60	876.827854	972.372146
1	2018-08-01	764.05	782.70	742.259530	823.140470
2	2018-09-01	914.60	1014.40	961.988076	1066.811924
3	2018-10-01	948.40	963.80	914.002472	1013.597528
4	2018-11-01	918.00	898.80	852.360886	945.239114
5	2018-12-01	841.20	849.80	805.892614	893.707386
6	2019-01-01	841.40	874.60	829.411250	919.788750
7	2019-02-01	913.20	891.20	845.153562	937.246438
8	2019-03-01	1026.80	804.40	762.838337	845.961663
9	2019-04-01	848.20	844.60	800.961287	888.238713
10	2019-05-01	889.30	922.85	875.168273	970.531727
11	2019-06-01	838.00	857.20	812.910271	901.489729

In [827]:

```
# melt combineSales_df so i can use the hue parameter later on seaborn
melted_df = combineSales_df.melt('transaction_month', var_name='cols',  valu

# convert transaction_month to string datatype so xticks won't display time
melted_df['transaction_month'] = melted_df['transaction_month'].astype("str"
```

In [828]:
```
1 melted_df
```

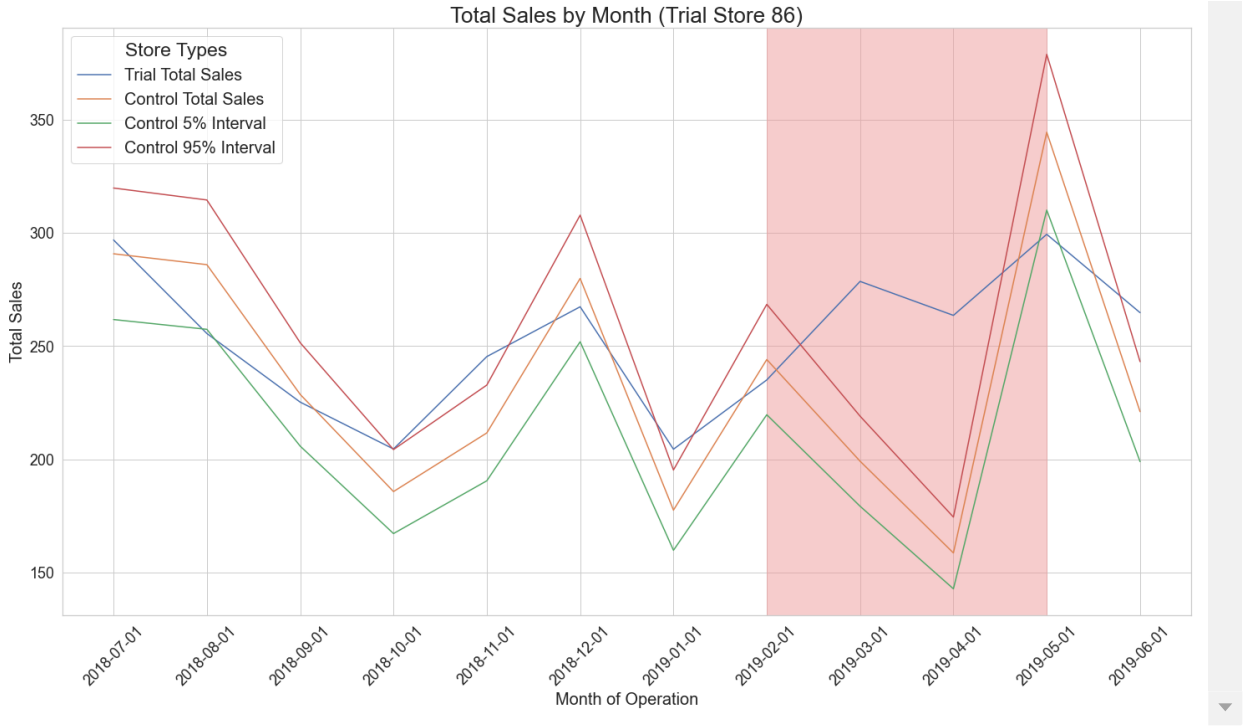
Out[828]:

	transaction_month	cols	vals
0	2018-07-01	trial_totSales	892.200000
1	2018-08-01	trial_totSales	764.050000
2	2018-09-01	trial_totSales	914.600000
3	2018-10-01	trial_totSales	948.400000
4	2018-11-01	trial_totSales	918.000000
5	2018-12-01	trial_totSales	841.200000
6	2019-01-01	trial_totSales	841.400000
7	2019-02-01	trial_totSales	913.200000
8	2019-03-01	trial_totSales	1026.800000
9	2019-04-01	trial_totSales	848.200000
10	2019-05-01	trial_totSales	889.300000
11	2019-06-01	trial_totSales	838.000000
12	2018-07-01	control_totSales	924.600000
13	2018-08-01	control_totSales	782.700000
14	2018-09-01	control_totSales	1014.400000
15	2018-10-01	control_totSales	963.800000
16	2018-11-01	control_totSales	898.800000
17	2018-12-01	control_totSales	849.800000
18	2019-01-01	control_totSales	874.600000
19	2019-02-01	control_totSales	891.200000
20	2019-03-01	control_totSales	804.400000
21	2019-04-01	control_totSales	844.600000
22	2019-05-01	control_totSales	922.850000
23	2019-06-01	control_totSales	857.200000
24	2018-07-01	control_5%_interval	876.827854
25	2018-08-01	control_5%_interval	742.259530
26	2018-09-01	control_5%_interval	961.988076
27	2018-10-01	control_5%_interval	914.002472
28	2018-11-01	control_5%_interval	852.360886
29	2018-12-01	control_5%_interval	805.892614
30	2019-01-01	control_5%_interval	829.411250
31	2019-02-01	control_5%_interval	845.153562
32	2019-03-01	control_5%_interval	762.838337
33	2019-04-01	control_5%_interval	800.961287

	transaction_month	cols	vals
34	2019-05-01	control_5%_interval	875.168273
35	2019-06-01	control_5%_interval	812.910271
36	2018-07-01	control_95%_interval	972.372146
37	2018-08-01	control_95%_interval	823.140470
38	2018-09-01	control_95%_interval	1066.811924
39	2018-10-01	control_95%_interval	1013.597528
40	2018-11-01	control_95%_interval	945.239114
41	2018-12-01	control_95%_interval	893.707386
42	2019-01-01	control_95%_interval	919.788750
43	2019-02-01	control_95%_interval	937.246438
44	2019-03-01	control_95%_interval	845.961663
45	2019-04-01	control_95%_interval	888.238713
46	2019-05-01	control_95%_interval	970.531727
47	2019-06-01	control_95%_interval	901.489729

In [933]:

```
# plot
sns.set(style="whitegrid")

# set figure size
plt.figure(figsize=(20,12))

# plot
g = sns.lineplot(x="transaction_month",
                 y="vals",
                 hue='cols',
                 data= melted_77_sales_df
                 )

# Highlight trial period
g.axvspan("2019-02-01", "2019-05-01", color='#EF9A9A', alpha=0.5)


# set x labels orientation
plt.xticks(rotation=45)

# # set fontsize
plt.xlabel("Month of Operation", fontsize= 20)
plt.ylabel("Total Sales", fontsize=20)
plt.title("Total Sales by Month (Trial Store 86)", fontsize=26)
plt.tick_params(labelsize=18)

legend_labels = ['Trial Total Sales',
                 "Control Total Sales",
                 "Control 5% Interval",
                 "Control 95% Interval"]

plt.legend(title = "Store Types",
           labels = legend_labels,
          loc='best',
          fontsize=20,
          title_fontsize=23)

plt.tight_layout()

# save figure
plt.savefig("static/module2_analysis_pics/trial_86_salesByMonth.png")
```



trial store 86 is not significantly different to its control store in the trial period. This relationship was shown in the graph where the trial store's performance lie within the confidence interval of the control store in two of the three trial months (trial months: from 2019-02 to 2019-06)

This time make a graph looking at the number of customers

In [836]:
```
1 print(trial_store)
2 print(ctrl_store)
```

```
86
155
```

In [837]:

```
### Find the Standard deviation based on scaled percentage difference in the

# scale pre_trial control sales to match pre-trial trial store sales

# trial store total sales
trial_nCustomers = store_data_df.loc[store_data_df['STORE_NBR'] == trial_sto

# control store total sales
ctrl_nCustomers = store_data_df.loc[store_data_df['STORE_NBR'] == ctrl_store

# scale numbers
scalingFactorForControlSales = trial_nCustomers / ctrl_nCustomers

# apply the scaling factor
scaled_control_stores = measureOverTime_df.loc[measureOverTime_df['STORE_NBR


# reorder columns
scaled_control_stores = scaled_control_stores[['STORE_NBR', 'yearmonth', 'nC
       'nChipsPerTxn', 'avgPricePerUnit', 'totSales']]

# add a customers feature
scaled_control_stores['control_nCustomers'] = scaled_control_stores['nCustom


## create a percentagedifference dataframe

# calcualte total sales for the trials data
trial_nCustomers = measureOverTime_df.loc[measureOverTime_df['STORE_NBR'] ==

# build the dataframe for analysis later
percentage_difference_df = scaled_control_stores[['yearmonth', 'control_nCus
percentage_difference_df['trial_nCustomers'] = trial_nCustomers.values


## calculate percentage difference

# select control_sales
control2_nCustomers = percentage_difference_df['control_nCustomers']

# select trial sales
trial2_nCustomers = percentage_difference_df['trial_nCustomers']

# store calculated value in a new column
percentage_difference_df['percentage_diff'] = abs(control2_nCustomers - tria

# reset index
percentage_difference_df.reset_index(drop=True, inplace=True)


# Calculate standard deviation based on the scaled percentage difference in
perDiff_preTrial = percentage_difference_df.loc[percentage_difference_df['ye
stdDev = perDiff_preTrial.std()
stdDev
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.p
```

```
y:33: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:45: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

Out[837]: 0.010687444701395238

In [838]:
```
measureOverTimeSales.head()
```

Out[838]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	t
0	1	201807	206.9	49	1.061224	1.192308	3.337097	
1	1	201808	176.1	42	1.023810	1.255814	3.261111	
2	1	201809	278.8	59	1.050847	1.209677	3.717333	
3	1	201810	188.1	44	1.022727	1.288889	3.243103	
4	1	201811	192.6	46	1.021739	1.212766	3.378947	

In [839]:
```
# trial and control store nCustomers
trial_nCustomers = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'
                                            ['transaction_month', 'nCustomers
control_nCustomers = measureOverTimeSales.loc[measureOverTimeSales['STORE_NB
                                              ['transaction_month', 'nCustome
```

In [840]:
```
display(trial_nCustomers.head())
display(control_nCustomers.head())
```

	transaction_month	nCustomers
0	2018-07-01	99
1	2018-08-01	94
2	2018-09-01	103
3	2018-10-01	109
4	2018-11-01	100

	transaction_month	nCustomers
0	2018-07-01	101
1	2018-08-01	91
2	2018-09-01	103
3	2018-10-01	108
4	2018-11-01	101

In [841]:
```
# rename columns for trial and control data
trial_nCustomers.columns = ['transaction_month', 'trial_nCustomers']
control_nCustomers.columns = ['transaction_month', 'control_nCustomers']
```

In [842]:
```
# merge the two dataframes
combineCustomers_df = pd.merge(trial_nCustomers, control_nCustomers, how='in
combineCustomers_df
```

Out[842]:

	transaction_month	trial_nCustomers	control_nCustomers
0	2018-07-01	99	101
1	2018-08-01	94	91
2	2018-09-01	103	103
3	2018-10-01	109	108
4	2018-11-01	100	101
5	2018-12-01	98	97
6	2019-01-01	94	96
7	2019-02-01	107	95
8	2019-03-01	115	94
9	2019-04-01	105	99
10	2019-05-01	104	106
11	2019-06-01	98	95

In [843]:
```
# create two columns marking the 95% and 5% confidence interval and graph la
combineCustomers_df['control_5%_interval'] = combineCustomers_df['control_nC

combineCustomers_df['control_95%_interval'] = combineCustomers_df['control_n
```

In [844]:
```
combineCustomers_df.head()
```

Out[844]:

	transaction_month	trial_nCustomers	control_nCustomers	control_5%_interval	control_95%_inter
0	2018-07-01	99	101	98.841136	103.1588
1	2018-08-01	94	91	89.054885	92.9451
2	2018-09-01	103	103	100.798386	105.2016
3	2018-10-01	109	108	105.691512	110.3084
4	2018-11-01	100	101	98.841136	103.1588

In [846]:
```
# melt combineCustomers_df for graphing
melted_86_customer_df = combineCustomers_df.melt('transaction_month',var_nam

# convert transaction month to string datatype so xticks won't display time
melted_86_customer_df['transaction_month'] = melted_86_customer_df['transact
```

In [847]:
```
display(melted_86_customer_df.head())
display(melted_86_customer_df.tail())
```

	transaction_month	cols	vals
0	2018-07-01	trial_nCustomers	99.0
1	2018-08-01	trial_nCustomers	94.0
2	2018-09-01	trial_nCustomers	103.0
3	2018-10-01	trial_nCustomers	109.0
4	2018-11-01	trial_nCustomers	100.0

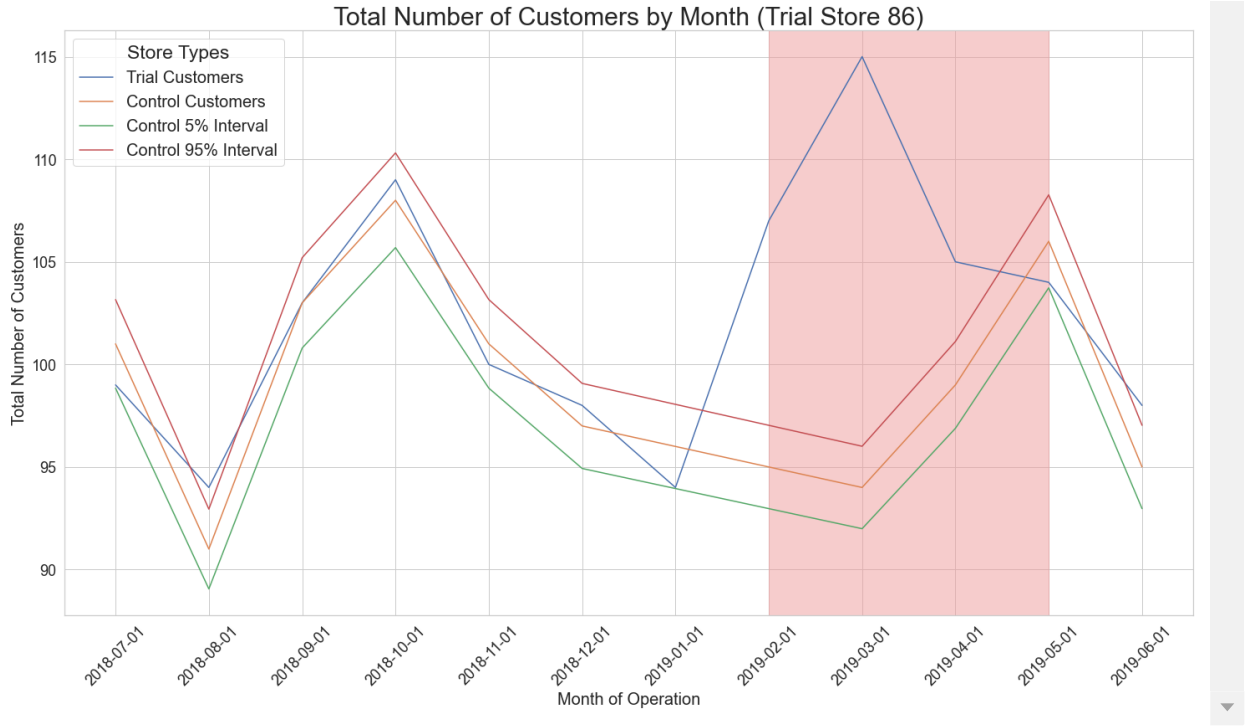
	transaction_month	cols	vals
43	2019-02-01	control_95%_interval	97.030614
44	2019-03-01	control_95%_interval	96.009240
45	2019-04-01	control_95%_interval	101.116114
46	2019-05-01	control_95%_interval	108.265738
47	2019-06-01	control_95%_interval	97.030614

In [849]:
```
trial_store
```

Out[849]: 86

In [1017]:

```
sns.set(style='whitegrid')

# set figure size
plt.figure(figsize=(20,12))

# plot
g = sns.lineplot(x="transaction_month",
                 y="vals",
                 hue='cols',
                 data= melted_86_customer_df
                 )

# Highlight trial period
g.axvspan("2019-02-01", "2019-05-01", color='#EF9A9A', alpha=0.5)

# set x labels orientation for visability
plt.xticks(rotation=45)

# set font sizes and labels
plt.xlabel("Month of Operation", fontsize = 20)
plt.ylabel("Total Number of Customers", fontsize = 20)
plt.title("Total Number of Customers by Month (Trial Store 86)",fontsize=30)
plt.tick_params(labelsize=18)

# set legend labels
legend_labels = ['Trial Customers',
                 "Control Customers",
                 "Control 5% Interval",
                 "Control 95% Interval"]

# set legend
plt.legend(title = 'Store Types',
           labels = legend_labels,
          loc = 'best',
          fontsize=20,
          title_fontsize=23)

plt.tight_layout()

# save figure
plt.savefig("static/module2_analysis_pics/trial_86_customerByMonth.png")
```




```
In [ ]:
1
```

```
In [ ]:
1
```

```
In [ ]:
1
```

```
In [ ]:
1
```

```
In [ ]:
1
```

Trial store 88

do the same thing like i did for the previous two stores

```
In [854]:
1 trial_store = 88
2 # find correlation for sales and customers using previously defined function
3 corr_nSales = findCorrelation(preTrialMeasures, trial_store, "totSales")
4 corr_nCustomers = findCorrelation(preTrialMeasures, trial_store, 'nCustomers
```

```
In [855]:
1 # compute magnitude with new trial store
2 magnitude_nSales = findDifference(preTrialMeasures, trial_store, 'totSales')
3 magnitude_nCustomers = findDifference(preTrialMeasures, trial_store, 'nCusto
```

In [856]:
```
1 corr_nSales
```

Out[856]:

	trial_store	other_stores	measure	correlation
0	88	1	totSales	0.813636
1	88	2	totSales	-0.067927
2	88	3	totSales	-0.507847
3	88	4	totSales	-0.745566
4	88	5	totSales	0.190330
...	...	...	...	...
255	88	268	totSales	-0.021429
256	88	269	totSales	-0.172578
257	88	270	totSales	-0.723272
258	88	271	totSales	-0.103037
259	88	272	totSales	-0.772772

260 rows × 4 columns

In [857]:
```
1 # concatenate the sales together and customers together
2 corrMag_sales_df = pd.merge(corr_nSales, magnitude_nSales, how='inner')
3 corrMag_customers_df = pd.merge(corr_nCustomers, magnitude_nCustomers, how='
4 corrMag_sales_df.head()
```

Out[857]:

	trial_store	other_stores	measure	correlation	difference
0	88	1	totSales	0.813636	0.548959
1	88	2	totSales	-0.067927	0.541212
2	88	3	totSales	-0.507847	0.458109
3	88	4	totSales	-0.745566	0.484447
4	88	5	totSales	0.190330	0.496409

In [858]:
```
1 # calculate weighted average of each row using correlation and difference
2 # use 50% weighting for now assuming the two variables are equally important
3 corr_weight = 0.5
4 # for the sales dataframe
5 corrMag_sales_df['corrMagAVG_sales'] = corr_weight * corrMag_sales_df['corre
6                                       corr_weight * corrMag_sales_df['differen
7 # for the customers dataframe
8 corrMag_customers_df['corrMagAVG_customers'] = corr_weight * corrMag_custome
9                                               corr_weight * corrMag_customers_df['
```

In [859]:
```
print("corrMag_sales_df")
display(corrMag_sales_df.head())
print("corrMag_customers_df")
corrMag_customers_df.head()
```

corrMag_sales_df

	trial_store	other_stores	measure	correlation	difference	corrMagAVG_sales
0	88	1	totSales	0.813636	0.548959	0.681297
1	88	2	totSales	-0.067927	0.541212	0.236643
2	88	3	totSales	-0.507847	0.458109	-0.024869
3	88	4	totSales	-0.745566	0.484447	-0.130559
4	88	5	totSales	0.190330	0.496409	0.343370

corrMag_customers_df

Out[859]:

	trial_store	other_stores	measure	correlation	difference	corrMagAVG_customers
0	88	1	nCustomers	0.305334	0.357143	0.331238
1	88	2	nCustomers	-0.452379	0.285714	-0.083332
2	88	3	nCustomers	0.522884	0.683673	0.603279
3	88	4	nCustomers	-0.361503	0.577922	0.108210
4	88	5	nCustomers	-0.025320	0.558442	0.266561

In [860]:
```
### FIND CONTROL STORE ###

# merge into one dataframe with just the corrMagAVG column
# pd.merge(corrMag_sales_df.reset_index(), corrMag_customers_df.reset_index(
control_store_df = pd.concat([corrMag_sales_df, corrMag_customers_df], axis=
control_store_df.head()
```

Out[860]:

	trial_store	trial_store	other_stores	other_stores	corrMagAVG_sales	corrMagAVG_customers
0	88	88	1	1	0.681297	0.331238
1	88	88	2	2	0.236643	-0.083332
2	88	88	3	3	-0.024869	0.603279
3	88	88	4	4	-0.130559	0.108210
4	88	88	5	5	0.343370	0.266561

In [861]:
```
# remove duplicate columns
control_store_df = control_store_df.loc[:, ~control_store_df.columns.duplica
```

In [862]:
```
# check if changes are made
control_store_df.head()
```

Out[862]:

	trial_store	other_stores	corrMagAVG_sales	corrMagAVG_customers
0	88	1	0.681297	0.331238
1	88	2	0.236643	-0.083332
2	88	3	-0.024869	0.603279
3	88	4	-0.130559	0.108210
4	88	5	0.343370	0.266561

In [863]:
```
# Store with the highest corrMagAVG score will be selected as the control st
control_store_df['control_score'] = 0.5 * (control_store_df['corrMagAVG_sale
                                           control_store_df['corrMagAVG_cust
```

In [864]:
```
control_store_df.head()
```

Out[864]:

	trial_store	other_stores	corrMagAVG_sales	corrMagAVG_customers	control_score
0	88	1	0.681297	0.331238	0.506268
1	88	2	0.236643	-0.083332	0.076655
2	88	3	-0.024869	0.603279	0.289205
3	88	4	-0.130559	0.108210	-0.011175
4	88	5	0.343370	0.266561	0.304965

In [868]:
```
# select the control store based on the highest score
some_sorted_df = control_store_df.sort_values(by='control_score', ascending=
display(some_sorted_df)
# save dataframe as html
some_sorted_df.to_html("html_formatted_dataframes/module_two_files/trial88Pr
```

	trial_store	other_stores	corrMagAVG_sales	corrMagAVG_customers	control_score
170	88	178	0.650803	0.707828	0.679316
12	88	14	0.646064	0.685774	0.665919
126	88	134	0.775084	0.540154	0.657619
225	88	237	0.451974	0.777235	0.614604
179	88	187	0.616752	0.594560	0.605656

From this chart, since store 178 has the highest score, we will select store 155 as our control store.

The control scores doesn't seem very high but we will try it anyways to see how it compare to the trial store

In [869]:
```
# create checkpoint for preTrialMeasures
store_data_df = preTrialMeasures.reset_index(drop=True)
store_data_df.head()
```

Out[869]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
0	1	201807	206.9	49	1.061224	1.192308	3.337097
1	1	201808	176.1	42	1.023810	1.255814	3.261111
2	1	201809	278.8	59	1.050847	1.209677	3.717333
3	1	201810	188.1	44	1.022727	1.288889	3.243103
4	1	201811	192.6	46	1.021739	1.212766	3.378947

In [870]:
```
# set trial and control stores
ctrl_store = 178

trial_store
```

Out[870]: 88

In [871]:
```
# select just the trial and control store
store_data_df.loc[(store_data_df['STORE_NBR'] == ctrl_store) |
                  (store_data_df['STORE_NBR'] == trial_store)]
```

Out[871]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
574	88	201807	1310.0	129	1.186047	2.000000	4.281046
575	88	201808	1323.8	131	1.221374	1.893750	4.368977
576	88	201809	1423.0	124	1.282258	2.000000	4.474843
577	88	201810	1352.4	123	1.284553	2.000000	4.279747
578	88	201811	1382.8	130	1.207692	2.000000	4.403822
579	88	201812	1325.2	126	1.182540	2.000000	4.446980
580	88	201901	1266.4	117	1.247863	2.000000	4.336986
1190	178	201807	952.0	107	1.233645	2.000000	3.606061
1191	178	201808	915.5	108	1.250000	1.874074	3.618577
1192	178	201809	954.4	101	1.316832	2.000000	3.587970
1193	178	201810	962.6	102	1.421569	2.000000	3.319310
1194	178	201811	975.6	111	1.252252	2.000000	3.509353
1195	178	201812	947.2	101	1.336634	2.000000	3.508148
1196	178	201901	837.2	95	1.284211	2.000000	3.431148

In [872]:
```
# create store types for graphing later (store type as in trial/control/othe
store_label = []
for i in range(store_data_df.shape[0]):
    if store_data_df['STORE_NBR'][i] == trial_store:
        store_label.append("trial_store")
    elif store_data_df['STORE_NBR'][i] == ctrl_store:
        store_label.append("control_store")
    else:
        store_label.append("other_stores")

store_data_df['store_type'] = store_label
```

In [873]:
```
# add a date column to store_data_df
store_data_df['transaction_dates'] = pd.to_datetime(store_data_df['yearmonth
```

In [874]:
```
store_data_df.head()
```

Out[874]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	
0	1	201807	206.9	49	1.061224	1.192308	3.337097	o
1	1	201808	176.1	42	1.023810	1.255814	3.261111	o
2	1	201809	278.8	59	1.050847	1.209677	3.717333	o
3	1	201810	188.1	44	1.022727	1.288889	3.243103	o
4	1	201811	192.6	46	1.021739	1.212766	3.378947	o

In [881]:
```
store_data_df['store_type'].unique()
```

Out[881]:
```
array(['other_stores', 'trial_store', 'control_store'], dtype=object)
```

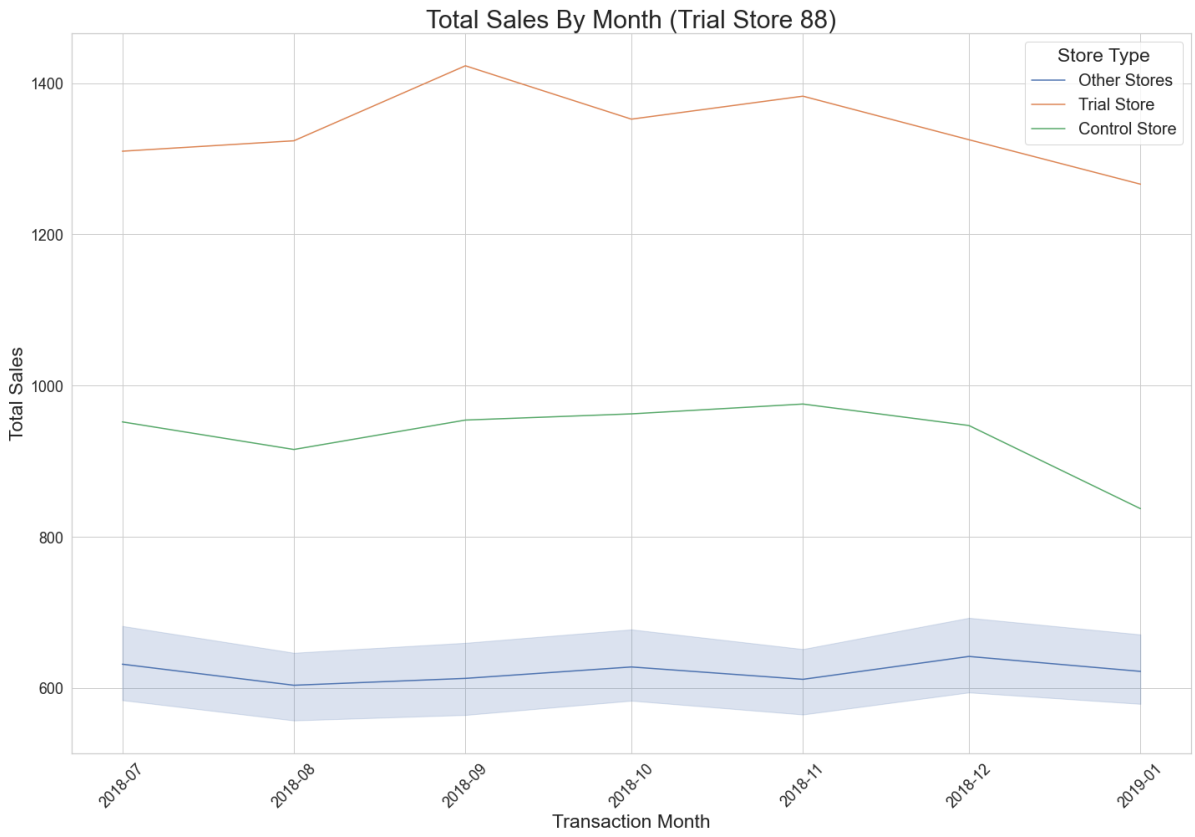
In [883]:
```
store_data_df.loc[store_data_df['store_type'] == "trial_store"]
```

Out[883]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
574	88	201807	1310.0	129	1.186047	2.00000	4.281046
575	88	201808	1323.8	131	1.221374	1.89375	4.368977
576	88	201809	1423.0	124	1.282258	2.00000	4.474843
577	88	201810	1352.4	123	1.284553	2.00000	4.279747
578	88	201811	1382.8	130	1.207692	2.00000	4.403822
579	88	201812	1325.2	126	1.182540	2.00000	4.446980
580	88	201901	1266.4	117	1.247863	2.00000	4.336986

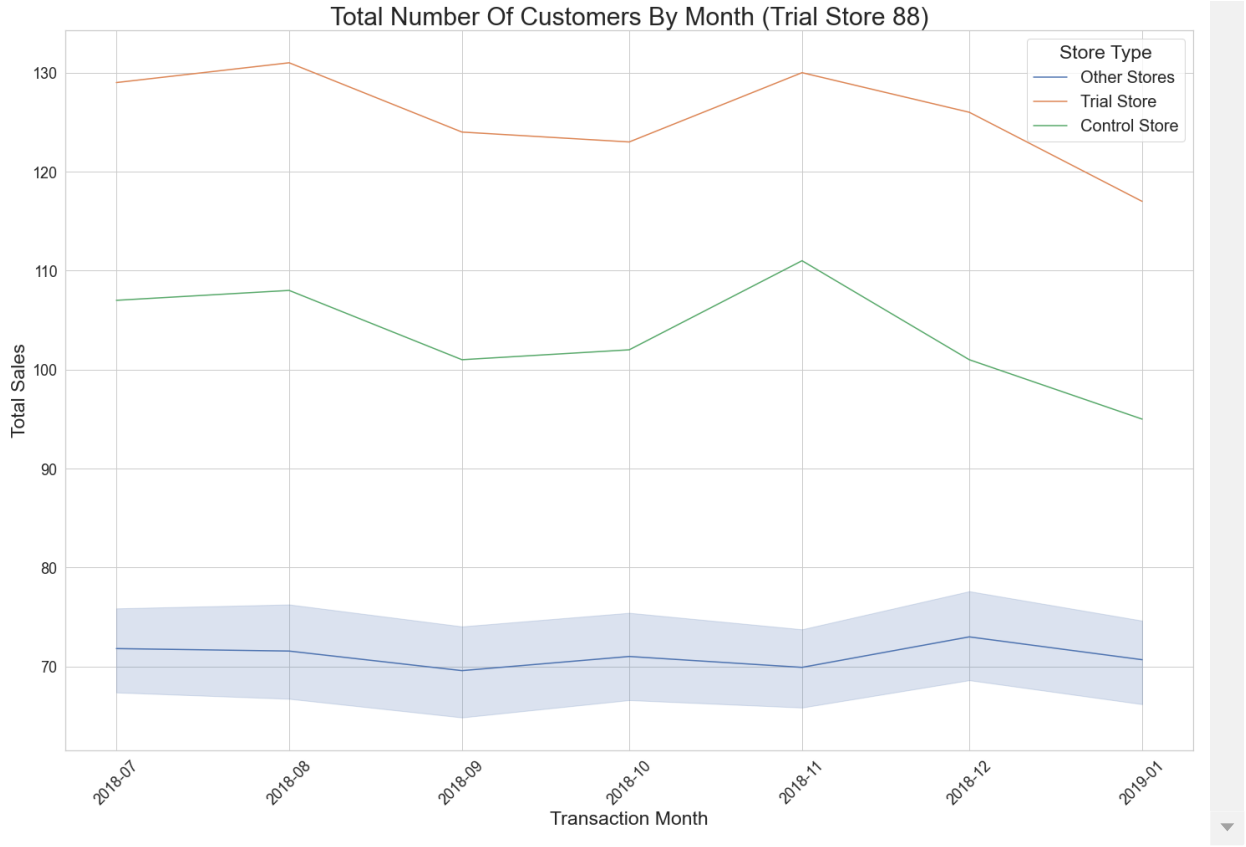
In [889]:

```
# visually check if drivers are indeed similar in the period before the tria
sns.set(style='whitegrid')
# set figure size
plt.figure(figsize=(20,14))

# create line chart
sns.lineplot(x = "transaction_dates",
             y = "totSales",
             hue = "store_type",
             data = store_data_df)

# set x label orientation
plt.xticks(rotation=45)

# set fontsize
plt.xlabel("Transaction Month", fontsize=23)
plt.ylabel("Total Sales", fontsize=23)
plt.title("total sales by month (trial store 88)".title(), fontsize=30)
plt.tick_params(labelsize=18)

# legend labels
some_labels = ['Other Stores',
               "Trial Store",
               "Control Store"]

# set legend location and fontsize
plt.legend(title = "store type".title(),
           labels = some_labels,
          loc='best',
          fontsize=20,
          title_fontsize=23)

# fit the graph
plt.tight_layout()

plt.savefig("static/module2_analysis_pics/trial_control_88_salesPerMonth.png
```


Total Sales By Month (Trial Store 88)
Store Type
Other Stores
Trial Store
Control Store
Total Sales
1400
1200
1000
800
600
2018-07
2018-08
2018-09
2018-10
2018-11
2018-12
2019-01
Transaction Month

In [888]:

```
# graph the customers data
# visually check if drivers are indeed similar in the period before the tria
sns.set(style='whitegrid')
# set figure size
plt.figure(figsize=(20,14))

# create line chart
sns.lineplot(x = "transaction_dates",
             y = "nCustomers",
             hue = "store_type",
             data = store_data_df)

# set x label orientation
plt.xticks(rotation=45)

# set fontsize
plt.xlabel("Transaction Month", fontsize=23)
plt.ylabel("Total Sales", fontsize=23)
plt.title("total number of customers by month (Trial Store 88)".title(), fon
plt.tick_params(labelsize=18)

# legend labels
some_labels = ['Other Stores',
               "Trial Store",
               "Control Store"]

# set legend location and fontsize
plt.legend(title = "store type".title(),
           labels=some_labels,
          loc='best',
          fontsize=20,
          title_fontsize=23)

# fit the graph
plt.tight_layout()

# save figure
plt.savefig("static/module2_analysis_pics/trial_control_88_customersPerMonth
```



Assessment for trial store 88

In [535]:
```
1 trial_store
```

Out[535]: 88

In [890]:
```
1  # scale pre_trial control sales to match pre-trial store sales
2
3  # trial store total sales
4  trial_totSales = store_data_df.loc[store_data_df['STORE_NBR'] == trial_store
5
6  # control store total sales
7  ctrl_totSales = store_data_df.loc[store_data_df['STORE_NBR'] == ctrl_store,
8
9  # scale numbers
10 scalingFactorForControlSales = trial_totSales / ctrl_totSales
11 scalingFactorForControlSales
```

Out[890]: 1.4338146535258616

In [891]:
```
# apply the scaling factor

scaled_control_stores = measureOverTime_df.loc[measureOverTime_df['STORE_NBR
scaled_control_stores
```

Out[891]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit
2069	178	201807	952.0	107	1.233645	2.000000	3.606061
2070	178	201808	915.5	108	1.250000	1.874074	3.618577
2071	178	201809	954.4	101	1.316832	2.000000	3.587970
2072	178	201810	962.6	102	1.421569	2.000000	3.319310
2073	178	201811	975.6	111	1.252252	2.000000	3.509353
2074	178	201812	947.2	101	1.336634	2.000000	3.508148
2075	178	201901	837.2	95	1.284211	2.000000	3.431148
2076	178	201902	1088.8	107	1.373832	2.000000	3.703401
2077	178	201903	998.8	114	1.254386	2.000000	3.492308
2078	178	201904	1059.8	117	1.282051	2.000000	3.532667
2079	178	201905	888.0	102	1.284314	1.908397	3.552000
2080	178	201906	722.6	92	1.173913	2.000000	3.345370

In [892]:
```
# reorder columns
scaled_control_stores = scaled_control_stores[['STORE_NBR', 'yearmonth', 'nC
       'nChipsPerTxn', 'avgPricePerUnit', 'totSales']]
```

In [893]:
```
scaled_control_stores['control_sales'] = scaled_control_stores['totSales'] *
```

In [894]:
```
1 scaled_control_stores
```

Out[894]:

	STORE_NBR	yearmonth	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	totSales
2069	178	201807	107	1.233645	2.000000	3.606061	952.0
2070	178	201808	108	1.250000	1.874074	3.618577	915.5
2071	178	201809	101	1.316832	2.000000	3.587970	954.4
2072	178	201810	102	1.421569	2.000000	3.319310	962.6
2073	178	201811	111	1.252252	2.000000	3.509353	975.6
2074	178	201812	101	1.336634	2.000000	3.508148	947.2
2075	178	201901	95	1.284211	2.000000	3.431148	837.2
2076	178	201902	107	1.373832	2.000000	3.703401	1088.8
2077	178	201903	114	1.254386	2.000000	3.492308	998.8
2078	178	201904	117	1.282051	2.000000	3.532667	1059.8
2079	178	201905	102	1.284314	1.908397	3.552000	888.0
2080	178	201906	92	1.173913	2.000000	3.345370	722.6

In [895]:

```
## create a percentagedifference dataframe

# calcualte total sales for the trials data
trial_sales = measureOverTime_df.loc[measureOverTime_df['STORE_NBR'] == tria

# build the dataframe for analysis later
percentage_difference_df = scaled_control_stores[['yearmonth', 'control_sale
percentage_difference_df['trial_sales'] = trial_sales.values
percentage_difference_df
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

Out[895]:

	yearmonth	control_sales	trial_sales
2069	201807	1364.991550	1310.00
2070	201808	1312.657315	1323.80
2071	201809	1368.432705	1423.00
2072	201810	1380.189985	1352.40
2073	201811	1398.829576	1382.80
2074	201812	1358.109240	1325.20
2075	201901	1200.389628	1266.40
2076	201902	1561.137395	1370.20
2077	201903	1432.094076	1477.20
2078	201904	1519.556770	1439.40
2079	201905	1273.227412	1308.25
2080	201906	1036.074469	1354.60

In [896]:
```
## calculate percentage difference

# select control_sales
control2_sales = percentage_difference_df['control_sales']

# select trial sales
trial2_sales = percentage_difference_df['trial_sales']

# store calculated value in a new column
percentage_difference_df['percentage_diff'] = abs(control2_sales - trial2_sa
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  # Remove the CWD from sys.path while we load stuff.
```

In [897]:
```
percentage_difference_df.reset_index(drop=True, inplace=True)
```

In [898]:
```
percentage_difference_df
```

Out[898]:

	yearmonth	control_sales	trial_sales	percentage_diff
0	201807	1364.991550	1310.00	0.040287
1	201808	1312.657315	1323.80	0.008489
2	201809	1368.432705	1423.00	0.039876
3	201810	1380.189985	1352.40	0.020135
4	201811	1398.829576	1382.80	0.011459
5	201812	1358.109240	1325.20	0.024232
6	201901	1200.389628	1266.40	0.054991
7	201902	1561.137395	1370.20	0.122307
8	201903	1432.094076	1477.20	0.031496
9	201904	1519.556770	1439.40	0.052750
10	201905	1273.227412	1308.25	0.027507
11	201906	1036.074469	1354.60	0.307435

Our null hypothesis is such that the trial period is the same as the pre-trial period Take the standard deviation based on the scaled percentage difference in the pre-trial period

In [899]:
```
perDiff_preTrial = percentage_difference_df.loc[percentage_difference_df['ye
stdDev = perDiff_preTrial.std()
stdDev
```

Out[899]: 0.01707405192279777

In [900]:
```
# test with null hypothesis of there being 0 difference between trial and co
percentage_difference_df['tValues'] = (percentage_difference_df['percentage_
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:
2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

In [901]:
```
# select the trial period
percentage_difference_df.loc[percentage_difference_df['yearmonth'] >= 201902
```

Out[901]:

	yearmonth	control_sales	trial_sales	percentage_diff	tValues
7	201902	1561.137395	1370.20	0.122307	7.163302
8	201903	1432.094076	1477.20	0.031496	1.844699
9	201904	1519.556770	1439.40	0.052750	3.089489
10	201905	1273.227412	1308.25	0.027507	1.611038
11	201906	1036.074469	1354.60	0.307435	18.005977

In [902]:
```
# find the 95th percentile of the t distribution with degrees of freedom (df
# 95% confidence level
# degree of freedom of 7 is derived from have 9 pre-trial months. dof = 8-1=
scipy.stats.t.ppf(q=0.95, df=7)
```

Out[902]: 1.894578605061305

In [903]:
```
# create another checkpoint so we don't mess up the original data
measureOverTimeSales = measureOverTime_df
```

In [904]:
```

# add a transaction month column
measureOverTimeSales['transaction_month'] = pd.to_datetime(measureOverTimeSa
```

In [905]:
```
# trial and control store total sales
trial_sales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'] ==
                                       ['transaction_month','totSales']].res

control_sales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'] =
                                        ['transaction_month','totSales']].re
```

In [906]:
```
display(trial_sales.head())
display(control_sales.head())
```

	transaction_month	totSales
0	2018-07-01	1310.0
1	2018-08-01	1323.8
2	2018-09-01	1423.0
3	2018-10-01	1352.4
4	2018-11-01	1382.8

	transaction_month	totSales
0	2018-07-01	952.0
1	2018-08-01	915.5
2	2018-09-01	954.4
3	2018-10-01	962.6
4	2018-11-01	975.6

In [907]:
```
# rename columns for trial and control data
trial_sales.columns = ['transaction_month', 'trial_totSales']
control_sales.columns = ['transaction_month', 'control_totSales']
```

In [908]:
```
combineSales_df = pd.merge(trial_sales, control_sales, how='inner')
combineSales_df
```

Out[908]:

	transaction_month	trial_totSales	control_totSales
0	2018-07-01	1310.00	952.0
1	2018-08-01	1323.80	915.5
2	2018-09-01	1423.00	954.4
3	2018-10-01	1352.40	962.6
4	2018-11-01	1382.80	975.6
5	2018-12-01	1325.20	947.2
6	2019-01-01	1266.40	837.2
7	2019-02-01	1370.20	1088.8
8	2019-03-01	1477.20	998.8
9	2019-04-01	1439.40	1059.8
10	2019-05-01	1308.25	888.0
11	2019-06-01	1354.60	722.6

In [909]:
```
# create two columns marking the 95% and 5% confidence interval for graphing
combineSales_df['control_5%_interval'] = combineSales_df['control_totSales']

combineSales_df['control_95%_interval'] = combineSales_df['control_totSales'

combineSales_df
```

Out[909]:

	transaction_month	trial_totSales	control_totSales	control_5%_interval	control_95%_interval
0	2018-07-01	1310.00	952.0	919.491005	984.508995
1	2018-08-01	1323.80	915.5	884.237411	946.762589
2	2018-09-01	1423.00	954.4	921.809050	986.990950
3	2018-10-01	1352.40	962.6	929.729035	995.470965
4	2018-11-01	1382.80	975.6	942.285110	1008.914890
5	2018-12-01	1325.20	947.2	914.854916	979.545084
6	2019-01-01	1266.40	837.2	808.611207	865.788793
7	2019-02-01	1370.20	1088.8	1051.619545	1125.980455
8	2019-03-01	1477.20	998.8	964.692874	1032.907126
9	2019-04-01	1439.40	1059.8	1023.609840	1095.990160
10	2019-05-01	1308.25	888.0	857.676484	918.323516
11	2019-06-01	1354.60	722.6	697.924580	747.275420

In [911]:

```
# melt combineSales_df so i can use the hue parameter later on seaborn
melted_88_sales_df = combineSales_df.melt('transaction_month', var_name='col

# convert transaction_month to string datatype so xticks won't display time
melted_88_sales_df['transaction_month'] = melted_88_sales_df['transaction_mo
```

In [912]:
```
1 melted_88_sales_df
```

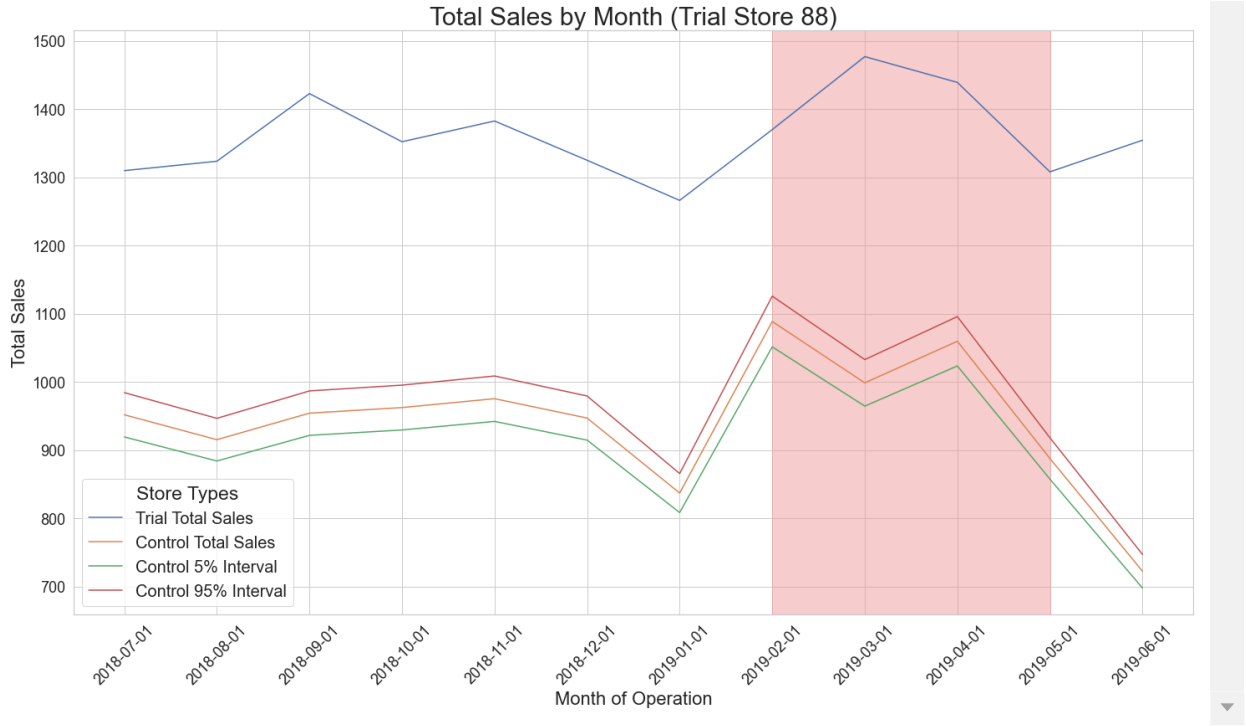
Out[912]:

	transaction_month	cols	vals
0	2018-07-01	trial_totSales	1310.000000
1	2018-08-01	trial_totSales	1323.800000
2	2018-09-01	trial_totSales	1423.000000
3	2018-10-01	trial_totSales	1352.400000
4	2018-11-01	trial_totSales	1382.800000
5	2018-12-01	trial_totSales	1325.200000
6	2019-01-01	trial_totSales	1266.400000
7	2019-02-01	trial_totSales	1370.200000
8	2019-03-01	trial_totSales	1477.200000
9	2019-04-01	trial_totSales	1439.400000
10	2019-05-01	trial_totSales	1308.250000
11	2019-06-01	trial_totSales	1354.600000
12	2018-07-01	control_totSales	952.000000
13	2018-08-01	control_totSales	915.500000
14	2018-09-01	control_totSales	954.400000
15	2018-10-01	control_totSales	962.600000
16	2018-11-01	control_totSales	975.600000
17	2018-12-01	control_totSales	947.200000
18	2019-01-01	control_totSales	837.200000
19	2019-02-01	control_totSales	1088.800000
20	2019-03-01	control_totSales	998.800000
21	2019-04-01	control_totSales	1059.800000
22	2019-05-01	control_totSales	888.000000
23	2019-06-01	control_totSales	722.600000
24	2018-07-01	control_5%_interval	919.491005
25	2018-08-01	control_5%_interval	884.237411
26	2018-09-01	control_5%_interval	921.809050
27	2018-10-01	control_5%_interval	929.729035
28	2018-11-01	control_5%_interval	942.285110
29	2018-12-01	control_5%_interval	914.854916
30	2019-01-01	control_5%_interval	808.611207
31	2019-02-01	control_5%_interval	1051.619545
32	2019-03-01	control_5%_interval	964.692874
33	2019-04-01	control_5%_interval	1023.609840

	transaction_month	cols	vals
34	2019-05-01	control_5%_interval	857.676484
35	2019-06-01	control_5%_interval	697.924580
36	2018-07-01	control_95%_interval	984.508995
37	2018-08-01	control_95%_interval	946.762589
38	2018-09-01	control_95%_interval	986.990950
39	2018-10-01	control_95%_interval	995.470965
40	2018-11-01	control_95%_interval	1008.914890
41	2018-12-01	control_95%_interval	979.545084
42	2019-01-01	control_95%_interval	865.788793
43	2019-02-01	control_95%_interval	1125.980455
44	2019-03-01	control_95%_interval	1032.907126
45	2019-04-01	control_95%_interval	1095.990160
46	2019-05-01	control_95%_interval	918.323516
47	2019-06-01	control_95%_interval	747.275420

In [930]:

```
# plot
sns.set(style="whitegrid")

plt.figure(figsize=(20,12))

# plot
g = sns.lineplot(x="transaction_month",
                 y="vals",
                 hue='cols',
                 data= melted_88_sales_df
                 )

# Highlight trial period
g.axvspan("2019-02-01", "2019-05-01", color='#EF9A9A', alpha=0.5)

# set x labels orientation
plt.xticks(rotation=45)

# # set fontsize
plt.xlabel("Month of Operation", fontsize= 22)
plt.ylabel("Total Sales", fontsize=22)
plt.title("Total Sales by Month (Trial Store 88)", fontsize=30)
plt.tick_params(labelsize=18)

legend_labels = ['Trial Total Sales',
                 "Control Total Sales",
                 "Control 5% Interval",
                 "Control 95% Interval"]

plt.legend(title = "Store Types",
           labels = legend_labels,
          loc='best',
          fontsize=20,
          title_fontsize=23)

plt.tight_layout()

# save figure
plt.savefig("static/module2_analysis_pics/trial_88_salesByMonth.png")
```



trial store 86 is not significantly different to its control store in the trial period. This relationship was shown in the graph where the trial store's performance lie within the confidence interval of the control store in two of the three trial months (trial months: from 2019-02 to 2019-06)

This time make a graph looking at the number of customers

In [915]:
```
1 print(trial_store)
2 print(ctrl_store)
```

```
88
178
```

In [916]:

```
### Find the Standard deviation based on scaled percentage difference in the

# scale pre_trial control sales to match pre-trial trial store sales

# trial store total sales
trial_nCustomers = store_data_df.loc[store_data_df['STORE_NBR'] == trial_sto

# control store total sales
ctrl_nCustomers = store_data_df.loc[store_data_df['STORE_NBR'] == ctrl_store

# scale numbers
scalingFactorForControlSales = trial_nCustomers / ctrl_nCustomers

# apply the scaling factor
scaled_control_stores = measureOverTime_df.loc[measureOverTime_df['STORE_NBR


# reorder columns
scaled_control_stores = scaled_control_stores[['STORE_NBR', 'yearmonth', 'nC
       'nChipsPerTxn', 'avgPricePerUnit', 'totSales']]

# add a customers feature
scaled_control_stores['control_nCustomers'] = scaled_control_stores['nCustom


## create a percentagedifference dataframe

# calcualte total sales for the trials data
trial_nCustomers = measureOverTime_df.loc[measureOverTime_df['STORE_NBR'] ==

# build the dataframe for analysis later
percentage_difference_df = scaled_control_stores[['yearmonth', 'control_nCus
percentage_difference_df['trial_nCustomers'] = trial_nCustomers.values


## calculate percentage difference

# select control_sales
control2_nCustomers = percentage_difference_df['control_nCustomers']

# select trial sales
trial2_nCustomers = percentage_difference_df['trial_nCustomers']

# store calculated value in a new column
percentage_difference_df['percentage_diff'] = abs(control2_nCustomers - tria

# reset index
percentage_difference_df.reset_index(drop=True, inplace=True)


# Calculate standard deviation based on the scaled percentage difference in
perDiff_preTrial = percentage_difference_df.loc[percentage_difference_df['ye
stdDev = perDiff_preTrial.std()
stdDev
```

```
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.p
```

```
y:33: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
C:\Users\smartestpersonalive\Anaconda3\lib\site-packages\ipykernel_launcher.py:45: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

Out[916]: 0.012417826272588403

In [917]:
```
measureOverTimeSales.head()
```

Out[917]:

	STORE_NBR	yearmonth	totSales	nCustomers	nTxnPerCust	nChipsPerTxn	avgPricePerUnit	t
0	1	201807	206.9	49	1.061224	1.192308	3.337097	
1	1	201808	176.1	42	1.023810	1.255814	3.261111	
2	1	201809	278.8	59	1.050847	1.209677	3.717333	
3	1	201810	188.1	44	1.022727	1.288889	3.243103	
4	1	201811	192.6	46	1.021739	1.212766	3.378947	

In [918]:
```
# trial and control store nCustomers
trial_nCustomers = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'
                                            ['transaction_month', 'nCustomers
control_nCustomers = measureOverTimeSales.loc[measureOverTimeSales['STORE_NB
                                              ['transaction_month', 'nCustome
```

In [919]:
```
display(trial_nCustomers.head())
display(control_nCustomers.head())
```

	transaction_month	nCustomers
0	2018-07-01	129
1	2018-08-01	131
2	2018-09-01	124
3	2018-10-01	123
4	2018-11-01	130

	transaction_month	nCustomers
0	2018-07-01	107
1	2018-08-01	108
2	2018-09-01	101
3	2018-10-01	102
4	2018-11-01	111

In [920]:
```
# rename columns for trial and control data
trial_nCustomers.columns = ['transaction_month', 'trial_nCustomers']
control_nCustomers.columns = ['transaction_month', 'control_nCustomers']
```

In [921]:
```
# merge the two dataframes
combineCustomers_df = pd.merge(trial_nCustomers, control_nCustomers, how='in
combineCustomers_df
```

Out[921]:

	transaction_month	trial_nCustomers	control_nCustomers
0	2018-07-01	129	107
1	2018-08-01	131	108
2	2018-09-01	124	101
3	2018-10-01	123	102
4	2018-11-01	130	111
5	2018-12-01	126	101
6	2019-01-01	117	95
7	2019-02-01	124	107
8	2019-03-01	134	114
9	2019-04-01	128	117
10	2019-05-01	128	102
11	2019-06-01	121	92

In [922]:
```
# create two columns marking the 95% and 5% confidence interval and graph la
combineCustomers_df['control_5%_interval'] = combineCustomers_df['control_nC

combineCustomers_df['control_95%_interval'] = combineCustomers_df['control_n
```

In [923]:
```
combineCustomers_df.head()
```

Out[923]:

	transaction_month	trial_nCustomers	control_nCustomers	control_5%_interval	control_95%_inter
0	2018-07-01	129	107	104.342585	109.6574
1	2018-08-01	131	108	105.317750	110.6822
2	2018-09-01	124	101	98.491599	103.5084
3	2018-10-01	123	102	99.466763	104.5332
4	2018-11-01	130	111	108.243243	113.7567

In [924]:
```
# melt combineCustomers_df for graphing
melted_88_customer_df = combineCustomers_df.melt('transaction_month',var_nam

# convert transaction month to string datatype so xticks won't display time
melted_88_customer_df['transaction_month'] = melted_88_customer_df['transact
```

In [925]:
```
display(melted_88_customer_df.head())
display(melted_88_customer_df.tail())
```

	transaction_month	cols	vals
0	2018-07-01	trial_nCustomers	129.0
1	2018-08-01	trial_nCustomers	131.0
2	2018-09-01	trial_nCustomers	124.0
3	2018-10-01	trial_nCustomers	123.0
4	2018-11-01	trial_nCustomers	130.0

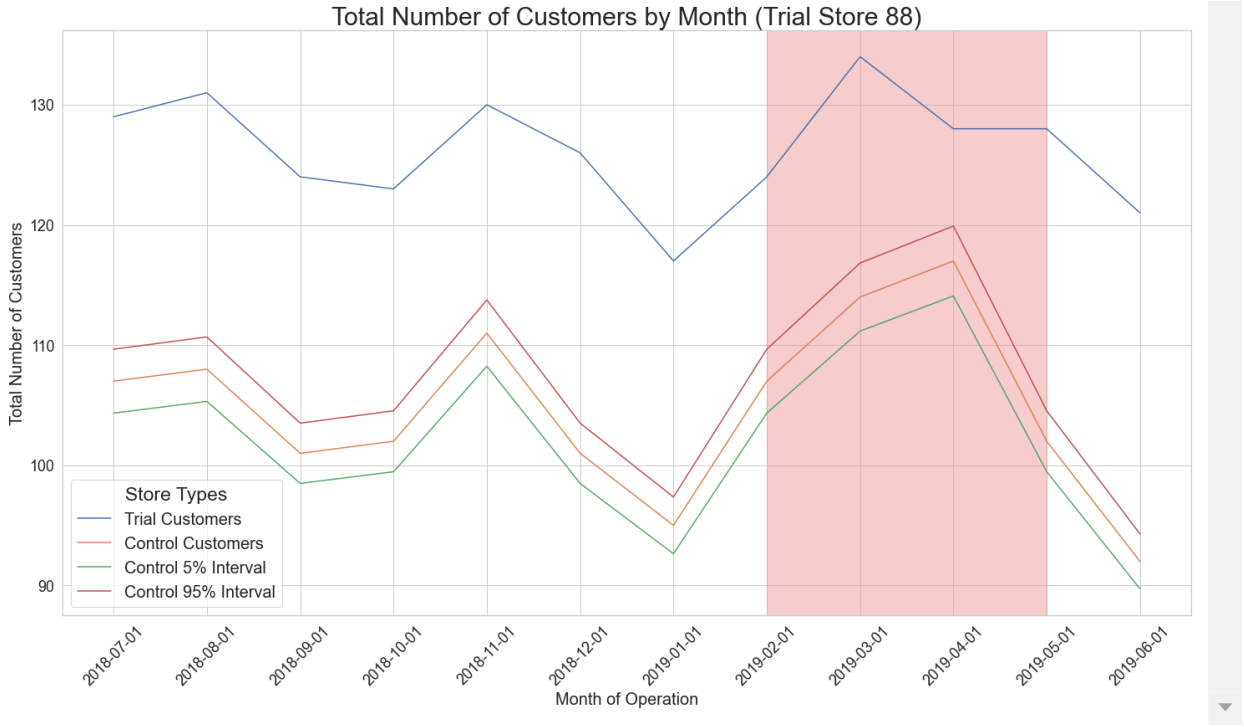
	transaction_month	cols	vals
43	2019-02-01	control_95%_interval	109.657415
44	2019-03-01	control_95%_interval	116.831264
45	2019-04-01	control_95%_interval	119.905771
46	2019-05-01	control_95%_interval	104.533237
47	2019-06-01	control_95%_interval	94.284880

In [926]:
```
trial_store
```

Out[926]: 88

In [1018]:

```
sns.set(style='whitegrid')

# set figure size
plt.figure(figsize=(20,12))

# plot
g = sns.lineplot(x="transaction_month",
                 y="vals",
                 hue='cols',
                 data= melted_88_customer_df
                 )

# Highlight trial period
g.axvspan("2019-02-01", "2019-05-01", color='#EF9A9A', alpha=0.5)

# set x labels orientation for visability
plt.xticks(rotation=45)

# set font sizes and labels
plt.xlabel("Month of Operation", fontsize = 20)
plt.ylabel("Total Number of Customers", fontsize = 20)
plt.title("Total Number of Customers by Month (Trial Store 88)",fontsize=30)
plt.tick_params(labelsize=18)

# set legend labels
legend_labels = ['Trial Customers',
                 "Control Customers",
                 "Control 5% Interval",
                 "Control 95% Interval"]

# set legend
plt.legend(title = 'Store Types',
           labels = legend_labels,
          loc = 'best',
          fontsize=20,
          title_fontsize=23)

plt.tight_layout()

plt.savefig("static/module2_analysis_pics/trial_88_customerByMonth.png")
```



In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```