Quantium Virtual Internship - Task 2

- examining the performance in trial vs control stores (77, 86 and 88) to provide a recommendation for each location
- exploring the data and define metrics for the control store selection
- checking each trial store individually in comparison with the control store to get a clear view of its overall performance and to know if the trial stores were successful or not
- Collate findings for each store and provide a recommendation on the impact on sales during the trial period

In [1]:
```
# Data analysis and wragling
import numpy as np
import pandas as pd

# Data manipulation and quality check
import datetime
import missingno as mn
import seaborn as sns

# Data visulaization
import matplotlib.pyplot as plt
import plotly
import plotly.express as px
import plotly.graph_objects as go

# for statistics
from scipy.stats import t

print('imported required libraries!')
```

```
imported required libraries!
```

In [2]:
```
# define a function to save plots

def saveplot(fig,fname):
    name='Plots/'+fname
    plotly.offline.plot(fig, filename=name)
```

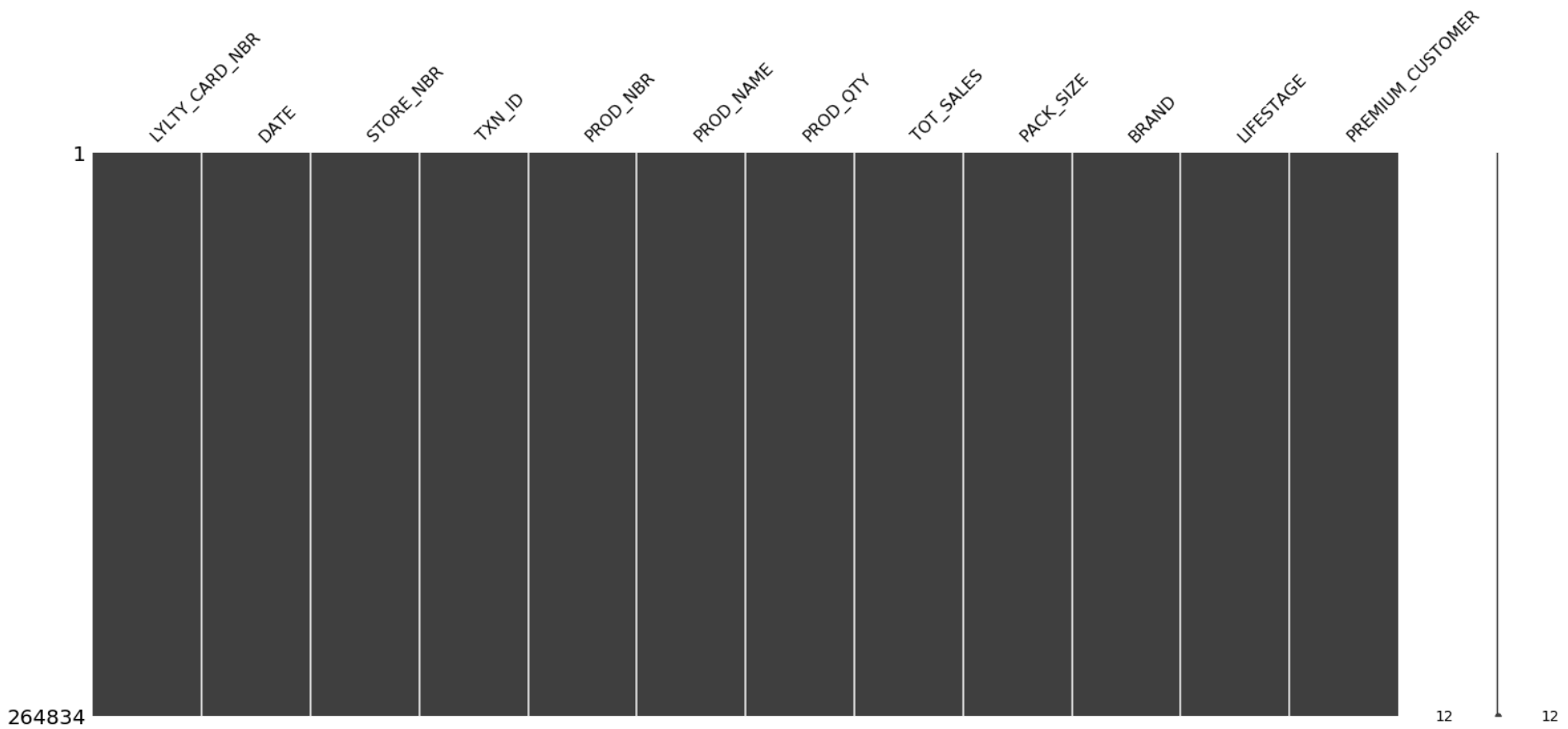
In [3]:
```
# Reading csv file and showing first 5 rows

df=pd.read_csv('Data/QVI_data.csv')
df.head(3)
```

Out[3]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NATURAL	YOUNG SINGLES/COUPLES	Premium
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	RRD	YOUNG SINGLES/COUPLES	Mainstream
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	GRNWVES	YOUNG FAMILIES	Budget

In [4]:
```
# checking, if there is any missing value in matrix
mn.matrix(df)
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19d4e867b88>
```



In [5]:
```
# getting info of the dataframe with column and row numbers, columns datatype and all

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]:
```
# getting statistical summary of the dataset

df.describe()
```

Out[6]:

	LYLTY_CARD_NBR	STORE_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES	PACK_SIZE
count	2.648340e+05	264834.000000	2.648340e+05	264834.000000	264834.000000	264834.000000	264834.000000
mean	1.355488e+05	135.079423	1.351576e+05	56.583554	1.905813	7.299346	182.425512
std	8.057990e+04	76.784063	7.813292e+04	32.826444	0.343436	2.527241	64.325148
min	1.000000e+03	1.000000	1.000000e+00	1.000000	1.000000	1.500000	70.000000
25%	7.002100e+04	70.000000	6.760050e+04	28.000000	2.000000	5.400000	150.000000
50%	1.303570e+05	130.000000	1.351365e+05	56.000000	2.000000	7.400000	170.000000
75%	2.030940e+05	203.000000	2.026998e+05	85.000000	2.000000	9.200000	175.000000
max	2.373711e+06	272.000000	2.415841e+06	114.000000	5.000000	29.500000	380.000000

In [7]:
```
# checking if there's any outliers,irrelevant or corrupt data

df['TOT_SALES'].value_counts().sort_values()
```

Out[7]:
```
11.2        2
12.4        2
9.3         3
15.5        3
6.9         3
          ...
8.8     19900
7.6     20212
6.0     20798
7.4     22513
9.2     22821
Name: TOT_SALES, Length: 111, dtype: int64
```

In [8]:
```
df['BRAND'].value_counts().sort_values()
```

Out[8]:
```
FRENCH         1418
BURGER         1564
CHEETOS        2927
SUNBITES       3008
CCS            4551
CHEEZELS       4603
TYRRELLS       6442
NATURAL        7469
GRNWVES        7740
OLD            9324
TWISTIES       9454
TOSTITOS       9471
COBS           9693
THINS         14075
INFUZIONS     14201
WOOLWORTHS    14757
RRD           17779
PRINGLES      25102
DORITOS       28145
SMITHS        31823
KETTLE        41288
Name: BRAND, dtype: int64
```

In [9]:
```
# inserting month column in the dataframe

df.insert(1, 'YEAR_MONTH',pd.to_datetime(df['DATE']).dt.to_period('M'))
df.head(3)
```

Out[9]:

	LYLTY_CARD_NBR	YEAR_MONTH	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	2018-10	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NATURAL	YOUNG SINGLES/COUPLES	Premium
1	1002	2018-09	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	RRD	YOUNG SINGLES/COUPLES	Mainstream
2	1003	2019-03	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	GRNWVES	YOUNG FAMILIES	Budget

In [10]:
```
# Looking for all the stores which don't contain transaction data of all 12 months

store = df.groupby('STORE_NBR')['YEAR_MONTH'].nunique()
store = store[store != 12]
print('Stores with less than 12 month transaction data:')
store=store.index.to_list()
store
```

```
Stores with less than 12 month transaction data:
```

Out[10]:
```
[11, 31, 44, 76, 85, 92, 117, 193, 206, 211, 218, 252]
```

In [11]:
```
# dropping those store that doesn't have trnsaction data of all 12 months

store_df=df[~df['STORE_NBR'].isin(store)]
store_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 264645 entries, 0 to 264833
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   LYLTY_CARD_NBR    264645 non-null  int64
 1   YEAR_MONTH        264645 non-null  period[M]
 2   DATE              264645 non-null  object
 3   STORE_NBR         264645 non-null  int64
 4   TXN_ID            264645 non-null  int64
 5   PROD_NBR          264645 non-null  int64
 6   PROD_NAME         264645 non-null  object
 7   PROD_QTY          264645 non-null  int64
 8   TOT_SALES         264645 non-null  float64
 9   PACK_SIZE         264645 non-null  int64
 10  BRAND             264645 non-null  object
 11  LIFESTAGE         264645 non-null  object
 12  PREMIUM_CUSTOMER  264645 non-null  object
dtypes: float64(1), int64(6), object(5), period[M](1)
memory usage: 28.3+ MB
```

Now we have got the dataset of all the stores with all 12 months transations

Let's first create the metrics of interest for each month and store, let's calculate:

1. Total sales (TOT_SALE)
2. Number of customers (TOT_CUST)
3. Transactions per customer (TRANS_CUST)
4. Product sale (chips) per transaction (PROD_TRANS)
5. Average price per unit (AVG_PRICE_UNIT)

In [12]:
```
# Defining a function to calculate metrics and create a new dataframe


def metrics(table):

    store_group = table.groupby(['STORE_NBR', 'YEAR_MONTH']) # grouping by store number and month

    total_sale = store_group['TOT_SALES'].sum() # calculating total sales in each store monthly

    num_cust = store_group['LYLTY_CARD_NBR'].nunique() # Number of customers in each store monthly

    transN = store_group['TXN_ID'].nunique() # number of transactions in each store monthly

    trans_cust = transN/ num_cust # number of transactions per customer in each store monthly

    avg_trans_per_cust = store_group['PROD_QTY'].sum() / transN # product sales per transaction in each store monthly

    avg_unit_price = total_sale / store_group['PROD_QTY'].sum() # average price per unit

    aggr = [total_sale, num_cust, trans_cust, avg_trans_per_cust, avg_unit_price] # creating a list of all column variables

    metrics = pd.concat(aggr, axis=1) # creating dataframe with all the calculated list

    metrics.columns = ['TOT_SALE','TOT_CUST','TRANS_CUST','PROD_TRANS','AVG_PRICE_UNIT'] # giving names to each column in dataframe

    return metrics # it will return dataframe named 'metrics'
```

In [13]:
```
# calling metrics function and reseting index in dataframe

store_metrics = metrics(store_df).reset_index()
store_metrics.head()
```

Out[13]:

	STORE_NBR	YEAR_MONTH	TOT_SALE	TOT_CUST	TRANS_CUST	PROD_TRANS	AVG_PRICE_UNIT
0	1	2018-07	206.9	49	1.061224	1.192308	3.337097
1	1	2018-08	176.1	42	1.023810	1.255814	3.261111
2	1	2018-09	278.8	59	1.050847	1.209677	3.717333
3	1	2018-10	188.1	44	1.022727	1.288889	3.243103
4	1	2018-11	192.6	46	1.021739	1.212766	3.378947

In [14]:
```
store_metrics.shape
```

Out[14]:
```
(3120, 7)
```

Pre-trial Period - before 2019-02

filter the stores that are present throughout the pre-trial period with the metrics

In [15]:
```
# Creating new dataframe 'pre_trial', which contains samples only before the trial period

pre_trial = store_metrics.loc[store_metrics['YEAR_MONTH'] < '2019-02', :]
pre_trial.head(3)
```

Out[15]:

	STORE_NBR	YEAR_MONTH	TOT_SALE	TOT_CUST	TRANS_CUST	PROD_TRANS	AVG_PRICE_UNIT
0	1	2018-07	206.9	49	1.061224	1.192308	3.337097
1	1	2018-08	176.1	42	1.023810	1.255814	3.261111
2	1	2018-09	278.8	59	1.050847	1.209677	3.717333

In [16]:
```
pre_trial.shape
```

Out[16]: (1820, 7)

Selecting Stores

The client has selected stores 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

First creating a function to calculate correlation for a measure, looping through each control store

Parameters:

- metric_col (str): Column names containing store's metric to perform correlation test
- store_comparison (int): Trial store's selcted number (77,86,88)
- input_table (dataframe): Metric table with comparison stores

---

Returns:
DataFrame: Monthly correlation table between Trial with each Control stores

In [17]:
```
def calcCorrTable(metric_col, store_comparison, input_table):

    # getting all the store numbers except selected for control stores
    ctrl_store_nbrs = input_table[~input_table['STORE_NBR'].isin([77, 86, 88])]['STORE_NBR'].unique()

    # creating a dataframe 'corrs' with column names defined in the list
    corrs = pd.DataFrame(columns =['YEAR_MONTH', 'Trial_Str', 'Control_Str', 'CORR_SCORE'])

    trial_store = input_table[input_table['STORE_NBR'] == store_comparison][metric_col].reset_index()

    for ctrl_stores in ctrl_store_nbrs:
        concat_df = pd.DataFrame(columns = ['YEAR_MONTH', 'Trial_Str', 'Control_Str', 'CORR_SCORE'])

        control_store = input_table[input_table['STORE_NBR'] == ctrl_stores][metric_col].reset_index()

        concat_df['CORR_SCORE'] = trial_store.corrwith(control_store, axis=1) # getting correlation score
        concat_df['Trial_Str'] = store_comparison # getting selected trial store number
        concat_df['Control_Str'] = ctrl_stores # getting control store number
        # getting month according to trail store number
        concat_df['YEAR_MONTH'] = list(input_table[input_table['STORE_NBR'] == store_comparison]['YEAR_MONTH'])

        # creating dataframe with correlated values
        corrs = pd.concat([corrs, concat_df])

    return corrs # returning final dataframe
```

Apart from correlation, we can also calculate a standardised metric based on the absolute difference between the trial store's performance and each control store's performance.

Create a function to calculate a standardised magnitude distance for a measure, looping through each control store

Parameters:

- metric_col (str): Column name containing store's metric to perform distance calculation
- store_comp (int): Trial store's number for comparison
- input_table (dataframe): Metric table with potential comparison stores.

Returns:
Dataframe: Monthly magnitude-distance table between trial and each Control stores

magnitude distance e.g. 1- (Observed distance – minimum distance)/(Maximum distance – minimum distance) as a measure

In [18]:
```
def calculateMagnitudeDistance(metric_col, store_comp, input_table):

    # getting all the store numbers except selected for control stores
    ctrl_store_nbrs = input_table[~input_table['STORE_NBR'].isin([77, 86, 88])]['STORE_NBR'].unique()


    # creating an empty dataframe 'dist'
    dist = pd.DataFrame()

    # getting trail store dataset
    trial_store = input_table[input_table['STORE_NBR'] == store_comp].reset_index()[metric_col]

    # looping through each control store
    for ctrl_str in ctrl_store_nbrs:

        # creating an empty dataframe to store all calculated values for trial stores
        calculated_df = pd.DataFrame()

        # getting absolute difference between the trial store's performance and each control store's performance
        calculated_df  = abs(trial_store - input_table[input_table['STORE_NBR'] == ctrl_str].reset_index()[metric_col])

        # getting year month according to trial store
        calculated_df['YEAR_MONTH'] = list(input_table[input_table['STORE_NBR'] == store_comp]['YEAR_MONTH'])

        # getting trial store numberinput_table[input_table['STORE_NBR'] == store_comp]['YEAR_MONTH'])
        calculated_df['Trial_Str'] = store_comp

        # getting control store number
        calculated_df['Control_Str'] = ctrl_str

        #joining dist with calculated dataframe
        dist = pd.concat([dist, calculated_df])

    for col in metric_col:
        dist[col] = 1 - ((dist[col] - dist[col].min()) / (dist[col].max() - dist[col].min()))

    dist['MAGNITUDE_DIST'] = dist[metric_col].mean(axis=1)

    # returning dataframe with the average value og magnitude distance for all trial stores compared with control stores
    return dist
```

Selecting control store for trial store (77, 86, 88)

In [19]:
```
#  A simple average on the scores would be :- 0.5 * corr_measure + 0.5 * mag_measure

corr_weight = 0.5
```

In [20]:

```
dist_table = pd.DataFrame()

for trial_store in [77, 86, 88]:

    # Compute correlation with trial store
    corr_nSales = calcCorrTable(['TOT_SALE'], trial_store, pre_trial)
    corr_nCustomers = calcCorrTable(['TOT_CUST'], trial_store, pre_trial)

    # Compute magnitude with trial store 86
    magnitude_nSales = calculateMagnitudeDistance(['TOT_SALE'], trial_store, pre_trial)
    magnitude_nCustomers = calculateMagnitudeDistance(['TOT_CUST'], trial_store, pre_trial)

    # Concatenate the scores together for 'nSales'
    score_nSales = pd.DataFrame()
    score_nSales = pd.concat([corr_nSales, magnitude_nSales['MAGNITUDE_DIST']], axis = 1)

    # Add an additional column which calculates the weighted average for sales
    score_nSales['scoreNSales'] = corr_weight * score_nSales['CORR_SCORE'] + (1 - corr_weight) * score_nSales['MAGNITUDE_DIST']

    # Concatenate the scores together for 'nCustomers'
    score_nCustomers = pd.DataFrame()
    score_nCustomers = pd.concat([corr_nCustomers, magnitude_nCustomers['MAGNITUDE_DIST']], axis = 1)

    # Add an additional column which calculates the weighted average for customer
    score_nCustomers['scoreNCust'] = corr_weight * score_nCustomers['CORR_SCORE'] + (1 - corr_weight) * score_nCustomers['MAGNITUDE_DIST']

    # Index both 'score_nSales' and 'score_nCustomers' dataframe

    score_nSales.set_index(['Trial_Str','Control_Str'], inplace = True)
    score_nCustomers.set_index(['Trial_Str','Control_Str'], inplace = True)

    # Create a new dataframe 'score_Control' which takes the average of 'scoreNSales' and 'scoreNCust'

    score_Control = pd.concat([score_nSales['scoreNSales'], score_nCustomers['scoreNCust']], axis = 1)

    # Add a new column to 'score_Control' which computes the average of 'scoreNSales' and 'scoreNCust'

    score_Control['finalControlScore'] = 0.5 * (score_Control['scoreNSales'] + score_Control['scoreNCust'])
    score_control = score_Control.sort_values(by = 'finalControlScore', ascending = False).head(1)
    dist_table = pd.concat([dist_table, score_control])

dist_table.reset_index(inplace=True)
dist_table
```

Out[20]:

	Trial_Str	Control_Str	scoreNSales	scoreNCust	finalControlScore
0	77	233	0.998779	1.000000	0.999389
1	86	225	0.998202	1.000000	0.999101
2	88	40	0.996386	0.992248	0.994317

shows the most correlated stores (trial store with control store)

We can see on the base of total Sale, number of customer and transaction per customer

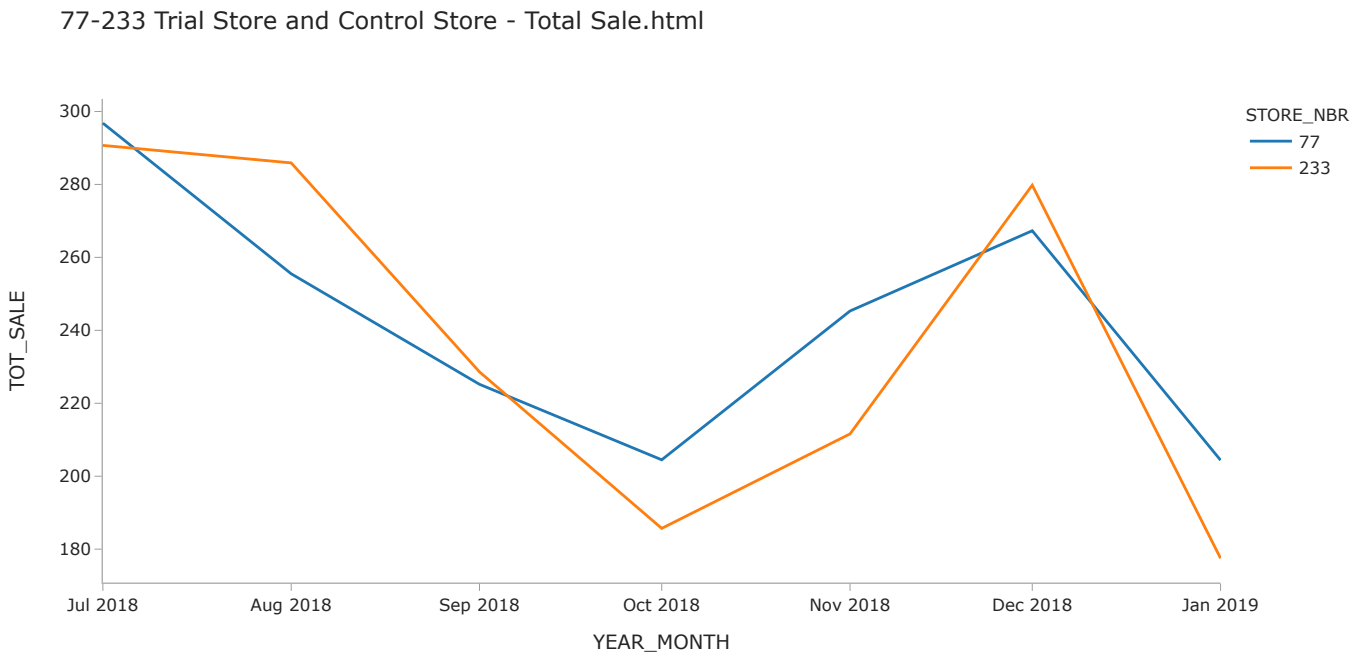
- Trial store 77 : 233 control store
- Trial store 86 : 225 control store
- Trial store 88 : 40 control store

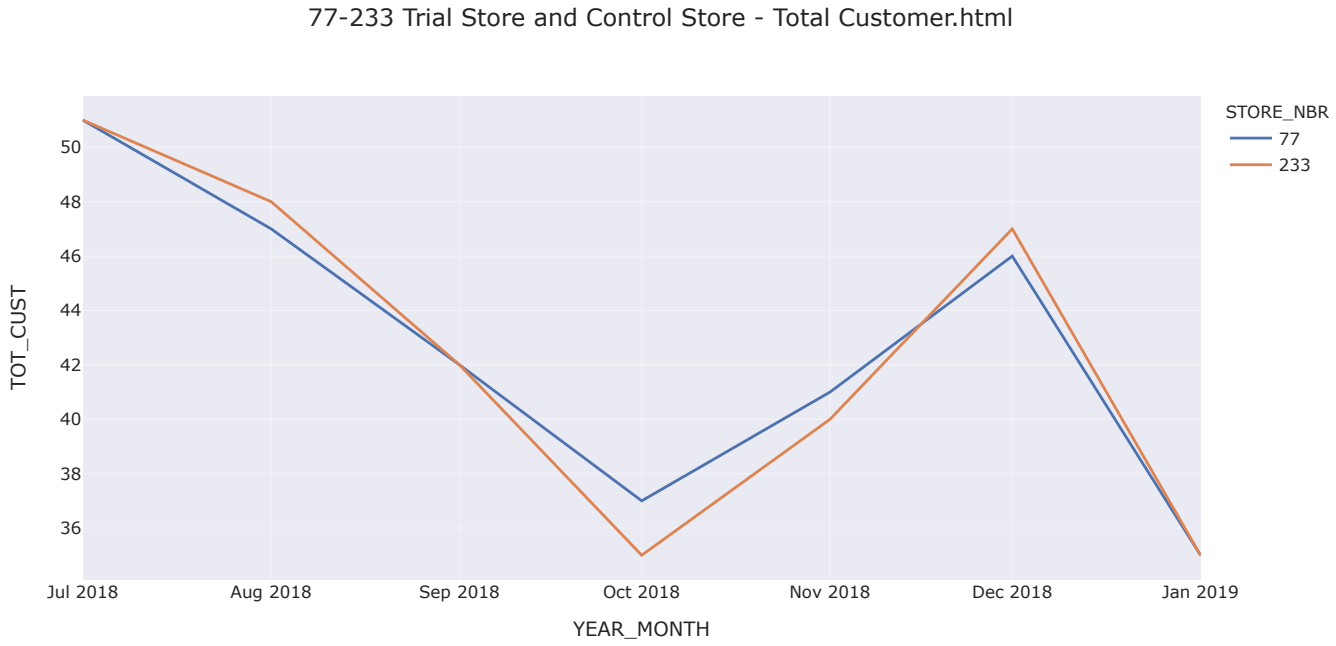
Visual checks on trends based on the drivers for pre-trial period

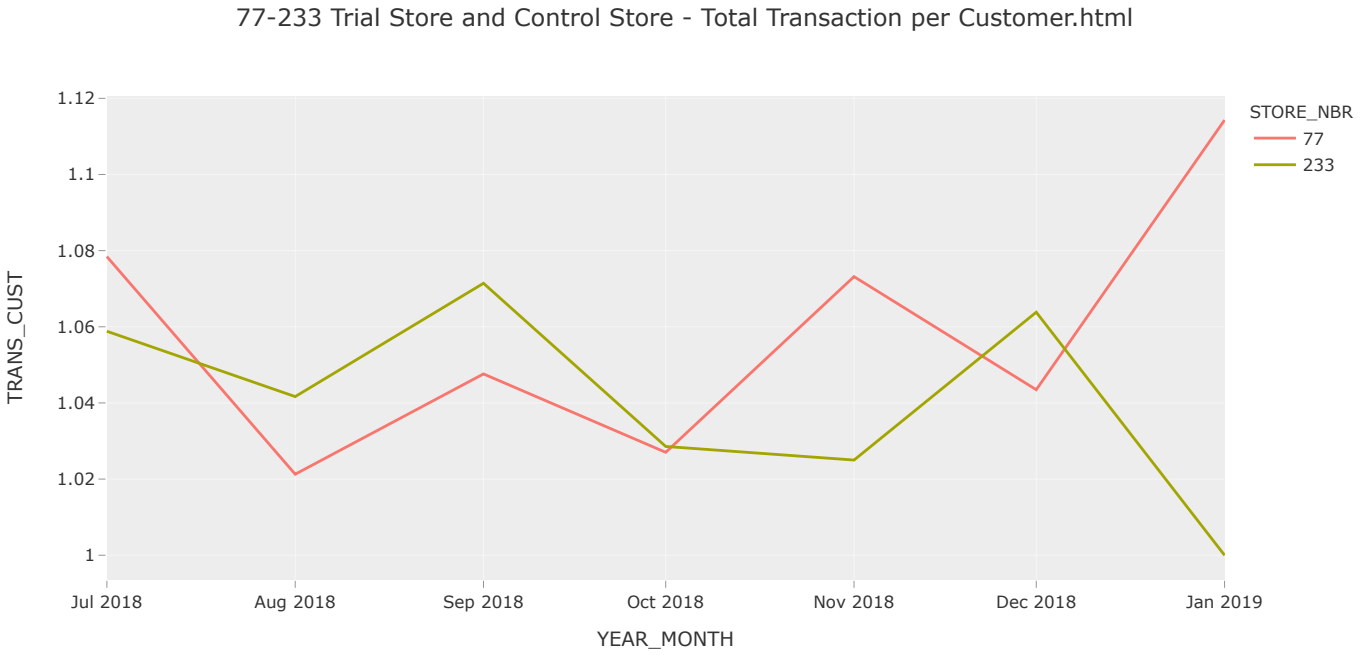
In [21]:

```
trial_control_dict={77:233,86:225,88:40}

for key, val in trial_control_dict.items():
    pre_trial['YEAR_MONTH']=pre_trial['YEAR_MONTH'].astype(str)

    matrix=pre_trial[pre_trial['STORE_NBR'].isin([key, val])].groupby(['YEAR_MONTH', 'STORE_NBR'])['TOT_SALE','TOT_CUST','TRANS_CUST','PROD_TRANS','AVG_PRICE_UNIT'].sum().reset
_index()

    name1=str(key)+'-'+str(val)+' Trial Store and Control Store - Total Sale.html'
    fig1 = px.line(matrix, x='YEAR_MONTH',y='TOT_SALE',color='STORE_NBR',template='simple_white',title=name1)
    saveplot(fig1,name1) # saving figure
    fig1.show()

    name2=str(key)+'-'+str(val)+' Trial Store and Control Store - Total Customer.html'
    fig2 = px.line(matrix, x='YEAR_MONTH',y='TOT_CUST',color='STORE_NBR',template='seaborn',title=name2)
    saveplot(fig2,name2) # saving figure
    fig2.show()

    name3=str(key)+'-'+str(val)+' Trial Store and Control Store - Total Transaction per Customer.html'
    fig3 = px.line(matrix, x='YEAR_MONTH',y='TRANS_CUST',color='STORE_NBR',template='ggplot2',title=name3)
    saveplot(fig3,name3) # saving figure
    fig3.show()

    name4=str(key)+'-'+str(val)+' Trial Store and Control Store - Total Chips purchase per Transaction.html'
    fig4 = px.line(matrix, x='YEAR_MONTH',y='PROD_TRANS',color='STORE_NBR',template='presentation',title=name4)
    saveplot(fig4,name4) # saving figure
    fig4.show()

    name5=str(key)+'-'+str(val)+' Trial Store and Control Store - Total Average Price per Unit.html'
    fig5 = px.line(matrix, x='YEAR_MONTH',y='AVG_PRICE_UNIT',color='STORE_NBR',template='simple_white',title=name5)
    saveplot(fig5,name5) # saving figure
    fig5.show()
```

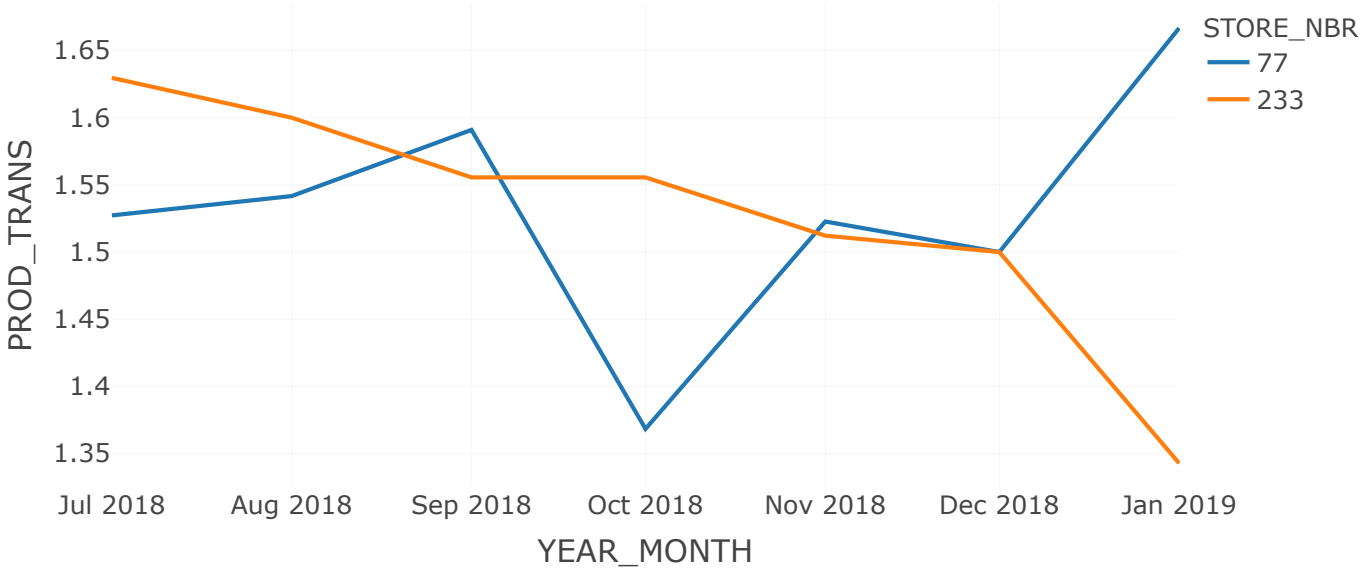
```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, us
e a list instead.
```



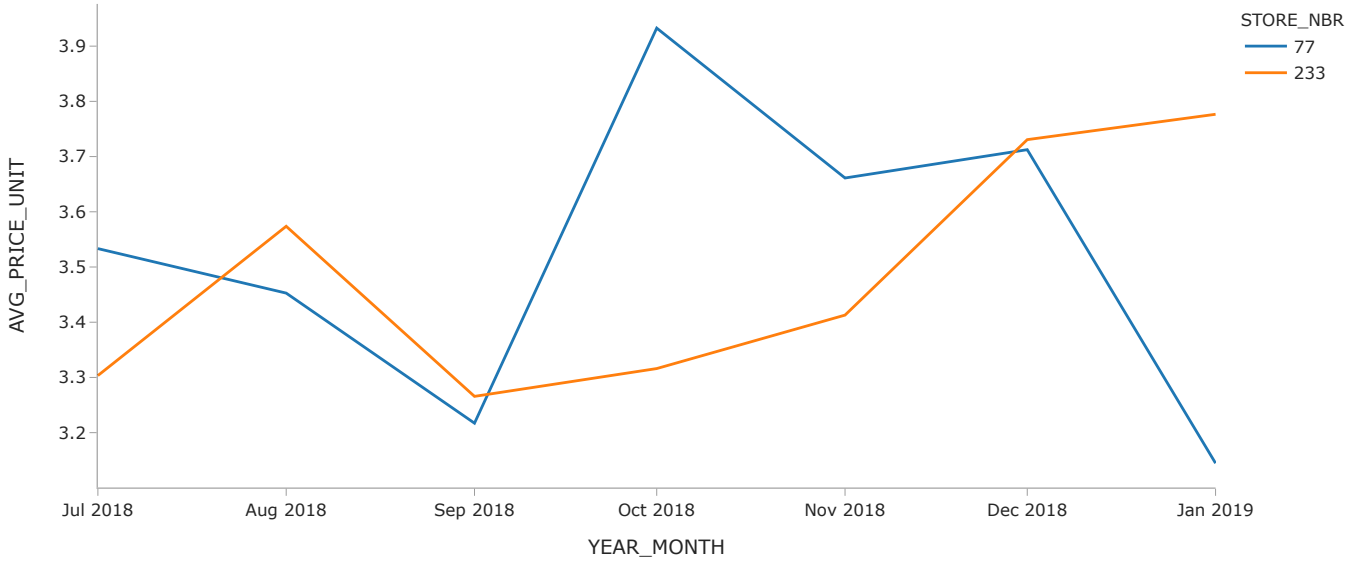

77-233 Trial Store and Control Store - Total Customer.html
TOT_CUST
50
48
46
44
42
40
38
36
Jul 2018
Aug 2018
Sep 2018
Oct 2018
Nov 2018
Dec 2018
Jan 2019
YEAR_MONTH
STORE_NBR
77
233


77-233 Trial Store and Control Store - Total Transaction per Customer.html
TRANS_CUST
1.12
1.1
1.08
1.06
1.04
1.02
1
Jul 2018
Aug 2018
Sep 2018
Oct 2018
Nov 2018
Dec 2018
Jan 2019
YEAR_MONTH
STORE_NBR
77
233

7-233 Trial Store and Control Store - Total Chips purchase per Transaction.htm

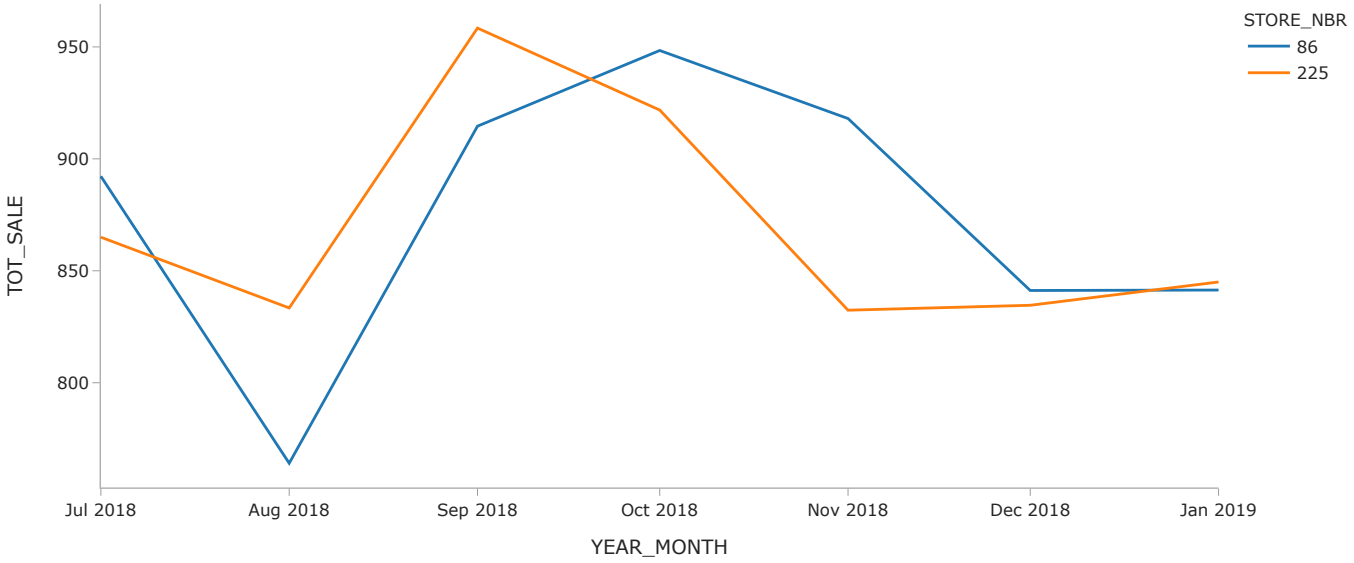


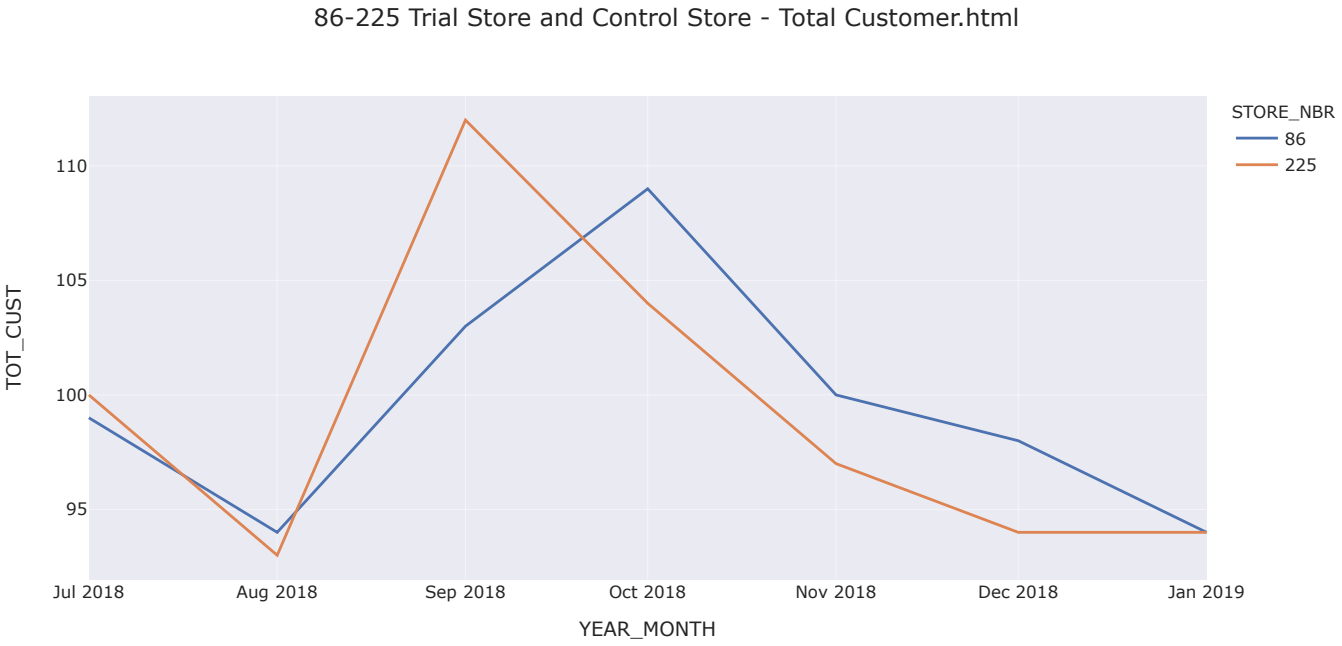
77-233 Trial Store and Control Store - Total Average Price per Unit.html



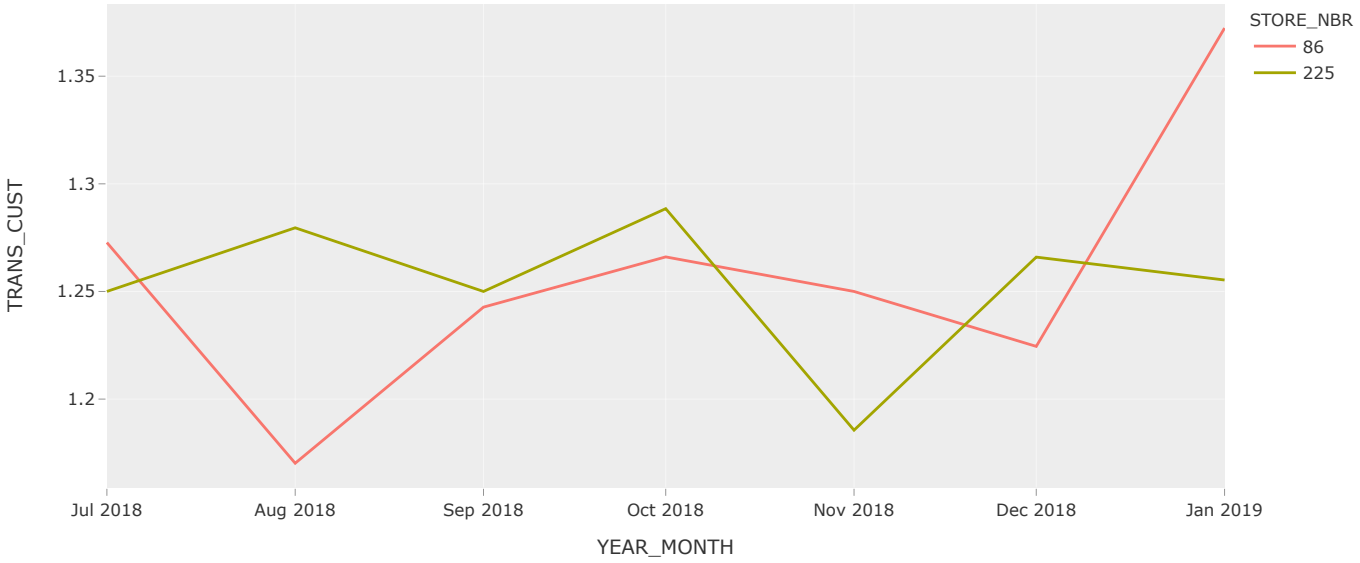

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:


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

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```

86-225 Trial Store and Control Store - Total Sale.html

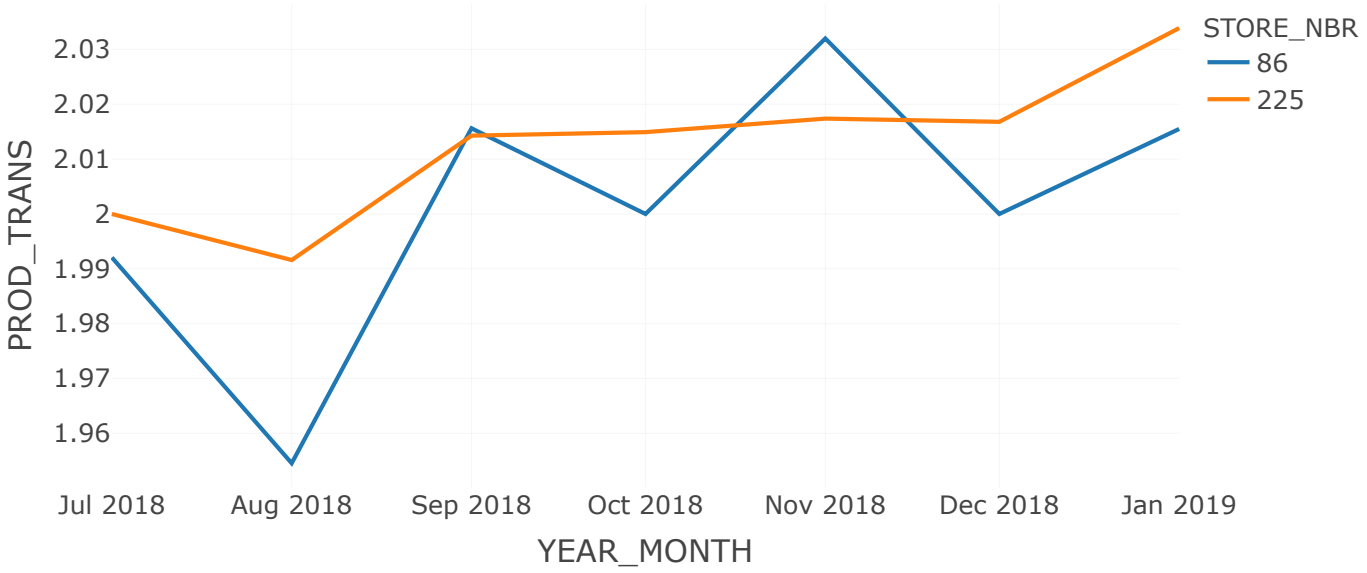



86-225 Trial Store and Control Store - Total Customer.html
STORE_NBR
86
225
TOT_CUST
110
105
100
95
Jul 2018
Aug 2018
Sep 2018
Oct 2018
Nov 2018
Dec 2018
Jan 2019
YEAR_MONTH

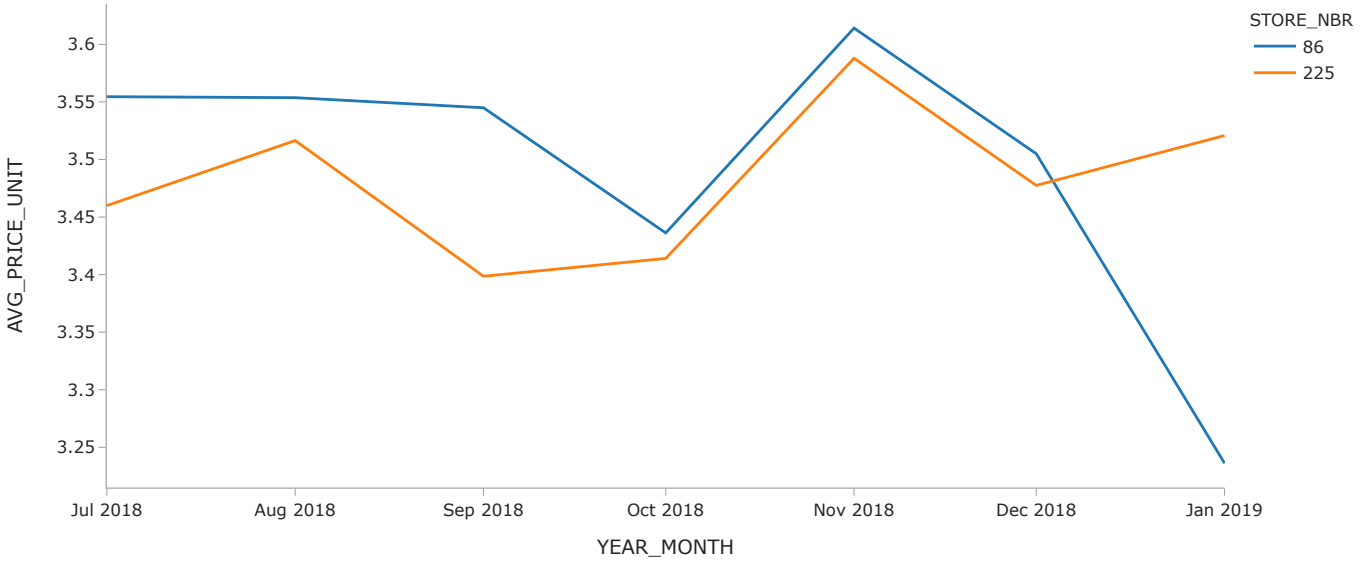
86-225 Trial Store and Control Store - Total Transaction per Customer.html



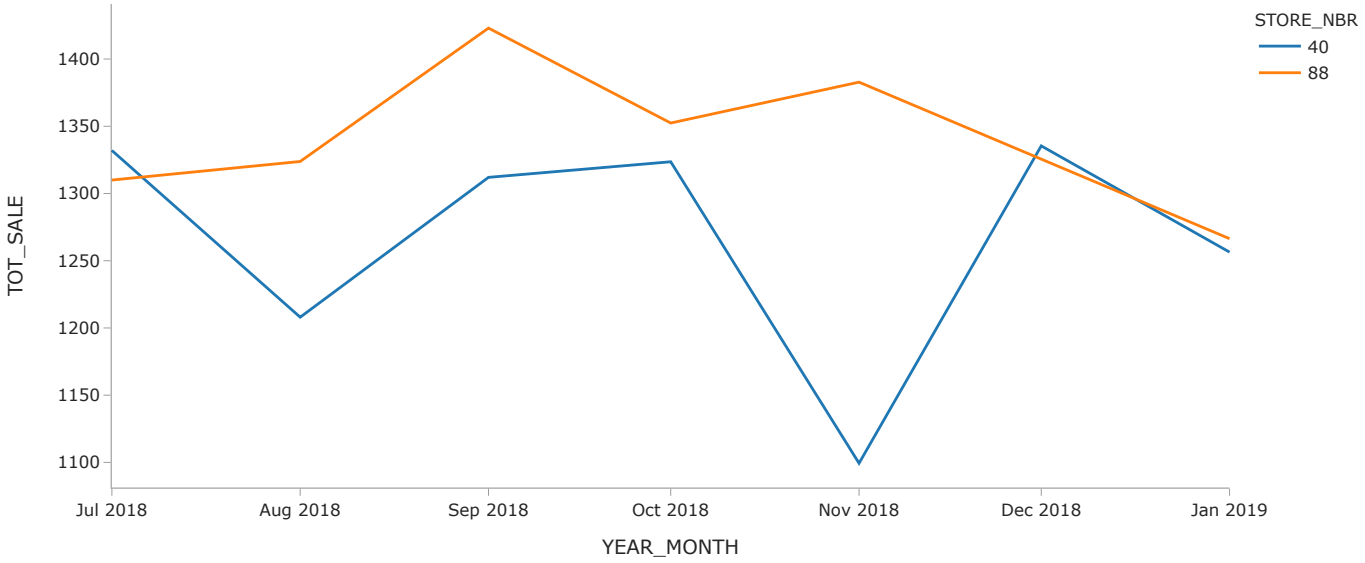
6-225 Trial Store and Control Store - Total Chips purchase per Transaction.htm

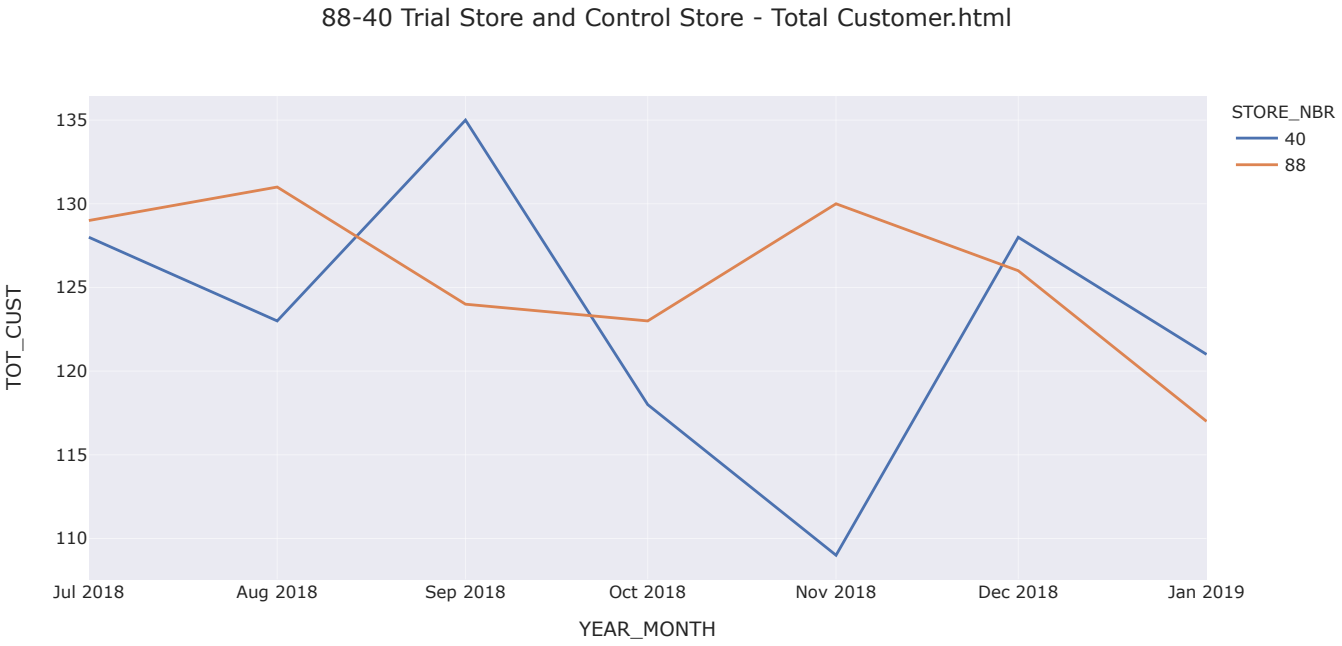


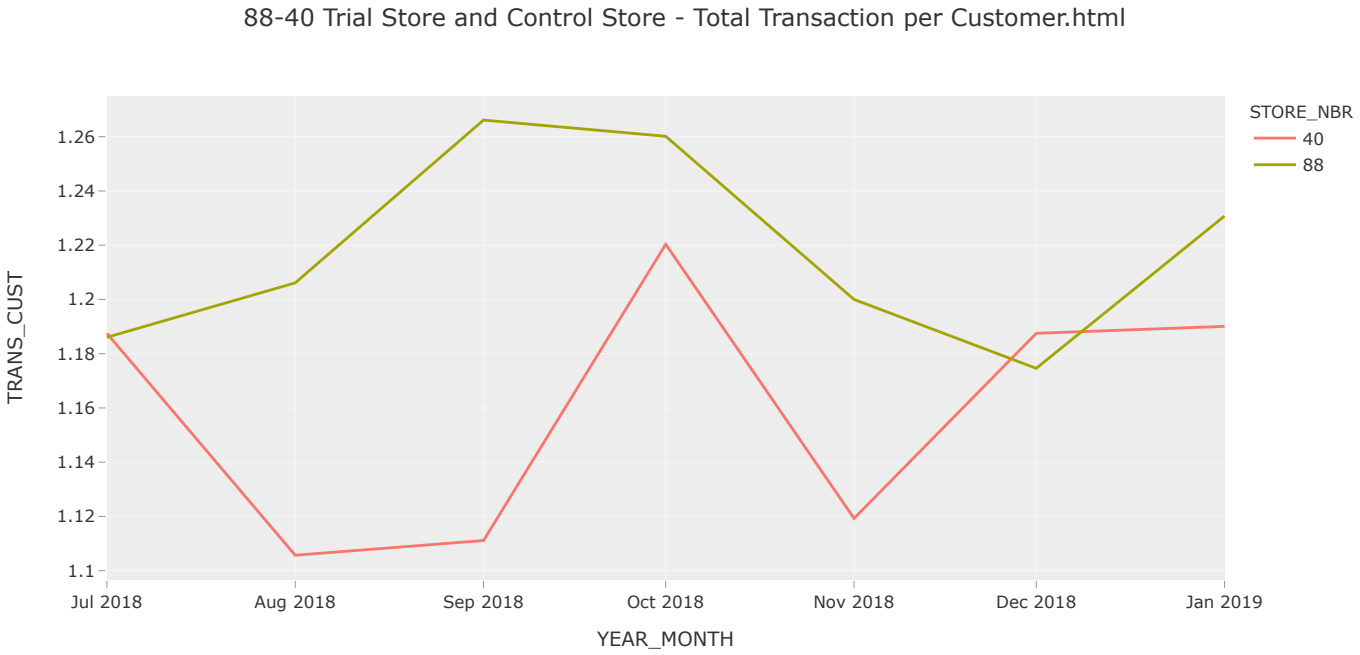
86-225 Trial Store and Control Store - Total Average Price per Unit.html

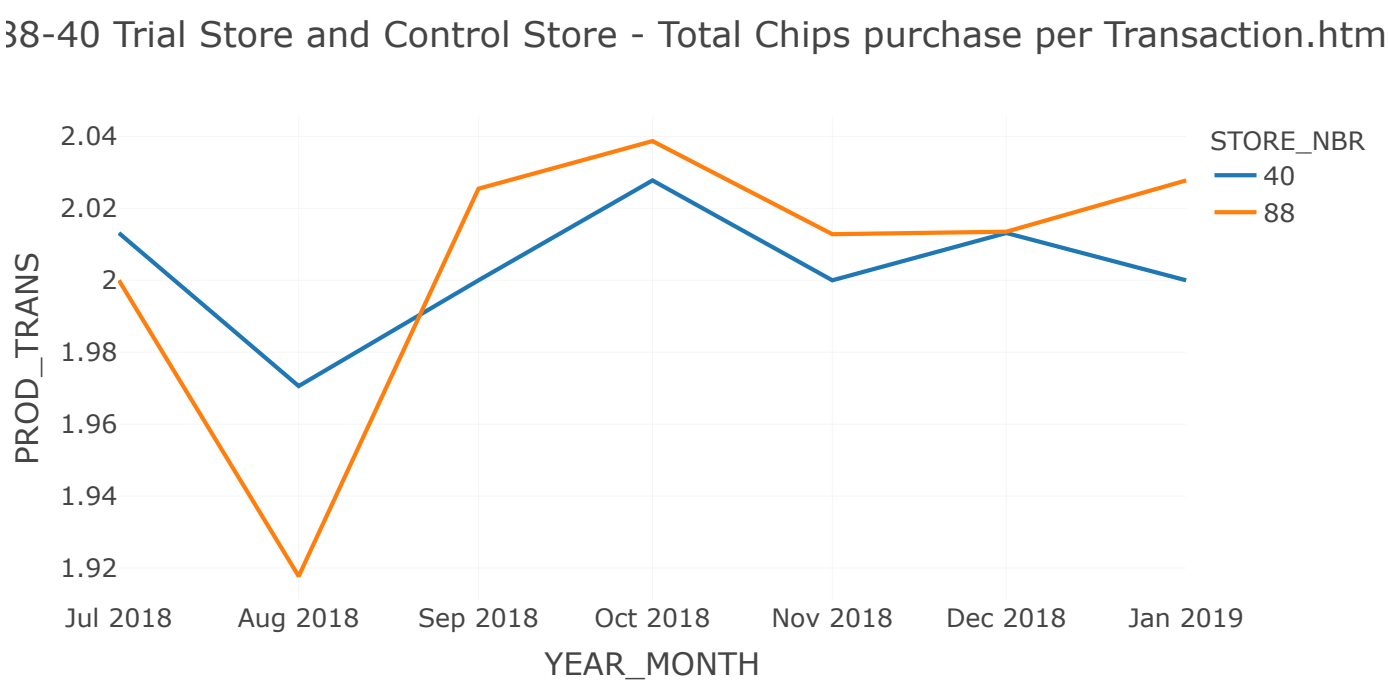


88-40 Trial Store and Control Store - Total Sale.html

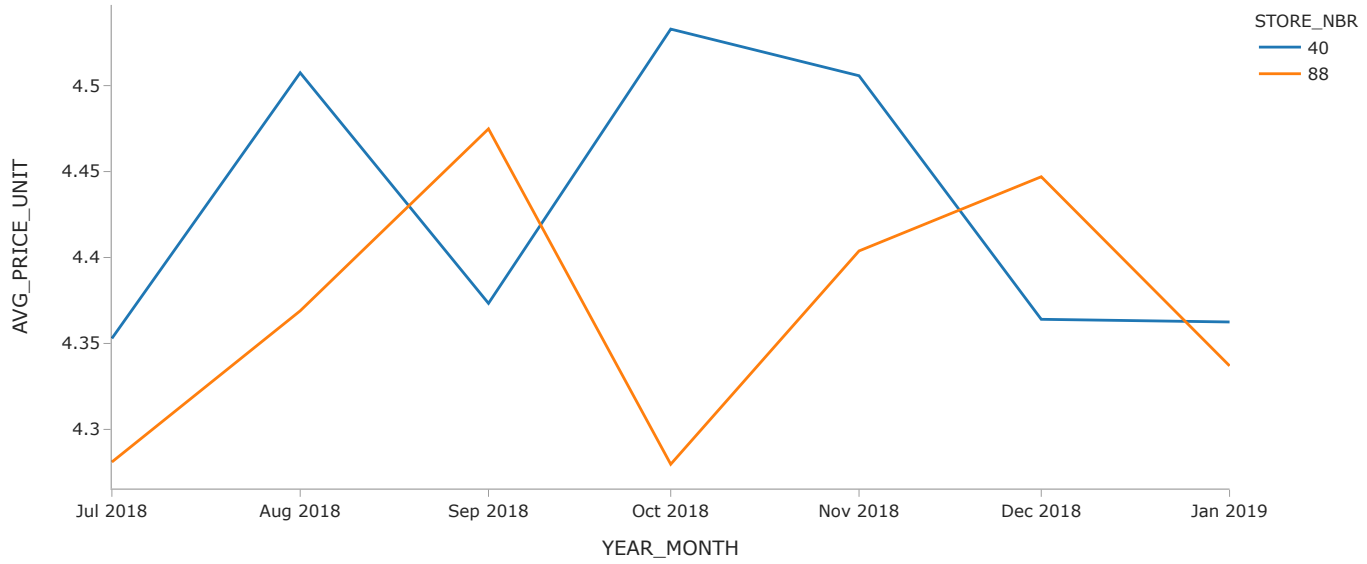



88-40 Trial Store and Control Store - Total Customer.html
TOT_CUST
135
130
125
120
115
110
Jul 2018
Aug 2018
Sep 2018
Oct 2018
Nov 2018
Dec 2018
Jan 2019
YEAR_MONTH
STORE_NBR
40
88


88-40 Trial Store and Control Store - Total Transaction per Customer.html
TRANS_CUST
1.26
1.24
1.22
1.2
1.18
1.16
1.14
1.12
1.1
Jul 2018
Aug 2018
Sep 2018
Oct 2018
Nov 2018
Dec 2018
Jan 2019
YEAR_MONTH
STORE_NBR
40
88


38-40 Trial Store and Control Store - Total Chips purchase per Transaction.htm
2.04
2.02
2
1.98
1.96
1.94
1.92
PROD_TRANS
Jul 2018
Aug 2018
Sep 2018
Oct 2018
Nov 2018
Dec 2018
Jan 2019
YEAR_MONTH
STORE_NBR
40
88

88-40 Trial Store and Control Store - Total Average Price per Unit.html



Now it is time to compare Trial stores to Control stores during the trial period.

Trial-period - From Feburary 2019 till April 2019

In [22]:
```
trial_metrics = store_metrics.loc[((store_metrics['YEAR_MONTH'] > '2019-01')& (store_metrics['YEAR_MONTH'] < '2019-05')) ,:]
trial_metrics.head(3)
```

Out[22]:

	STORE_NBR	YEAR_MONTH	TOT_SALE	TOT_CUST	TRANS_CUST	PROD_TRANS	AVG_PRICE_UNIT
7	1	2019-02	225.4	52	1.057692	1.181818	3.467692
8	1	2019-03	192.9	45	1.088889	1.183673	3.325862
9	1	2019-04	192.9	42	1.023810	1.325581	3.384211

In [23]:
```
dist_trial_table = pd.DataFrame()

for trial_store in [77, 86, 88]:

    # Compute correlation with trial store
    corr_nSales = calcCorrTable(['TOT_SALE'], trial_store, trial_metrics)
    corr_nCustomers = calcCorrTable(['TOT_CUST'], trial_store, trial_metrics)

    # Compute magnitude with trial store 86
    magnitude_nSales = calculateMagnitudeDistance(['TOT_SALE'], trial_store, trial_metrics)
    magnitude_nCustomers = calculateMagnitudeDistance(['TOT_CUST'], trial_store, trial_metrics)

    # Concatenate the scores together for 'nSales'
    score_nSales = pd.DataFrame()
    score_nSales = pd.concat([corr_nSales, magnitude_nSales['MAGNITUDE_DIST']], axis = 1)

    # Add an additional column which calculates the weighted average for sales
    score_nSales['scoreNSales'] = corr_weight * score_nSales['CORR_SCORE'] + (1 - corr_weight) * score_nSales['MAGNITUDE_DIST']

    # Concatenate the scores together for 'nCustomers'
    score_nCustomers = pd.DataFrame()
    score_nCustomers = pd.concat([corr_nCustomers, magnitude_nCustomers['MAGNITUDE_DIST']], axis = 1)

    # Add an additional column which calculates the weighted average for customer
    score_nCustomers['scoreNCust'] = corr_weight * score_nCustomers['CORR_SCORE'] + (1 - corr_weight) * score_nCustomers['MAGNITUDE_DIST']

    # Index both 'score_nSales' and 'score_nCustomers' dataframe

    score_nSales.set_index(['Trial_Str','Control_Str'], inplace = True)
    score_nCustomers.set_index(['Trial_Str','Control_Str'], inplace = True)

    # Create a new dataframe 'score_Control' which takes the average of 'scoreNSales' and 'scoreNCust'

    score_Control = pd.concat([score_nSales['scoreNSales'], score_nCustomers['scoreNCust']], axis = 1)

    # Add a new column to 'score_Control' which computes the average of 'scoreNSales' and 'scoreNCust'

    score_Control['finalControlScore'] = 0.5 * (score_Control['scoreNSales'] + score_Control['scoreNCust'])
    score_control = score_Control.sort_values(by = 'finalControlScore', ascending = False).head(1)
    dist_trial_table = pd.concat([dist_table, score_control])

dist_trial_table.reset_index(inplace=True)
dist_trial_table
```

Out[23]:

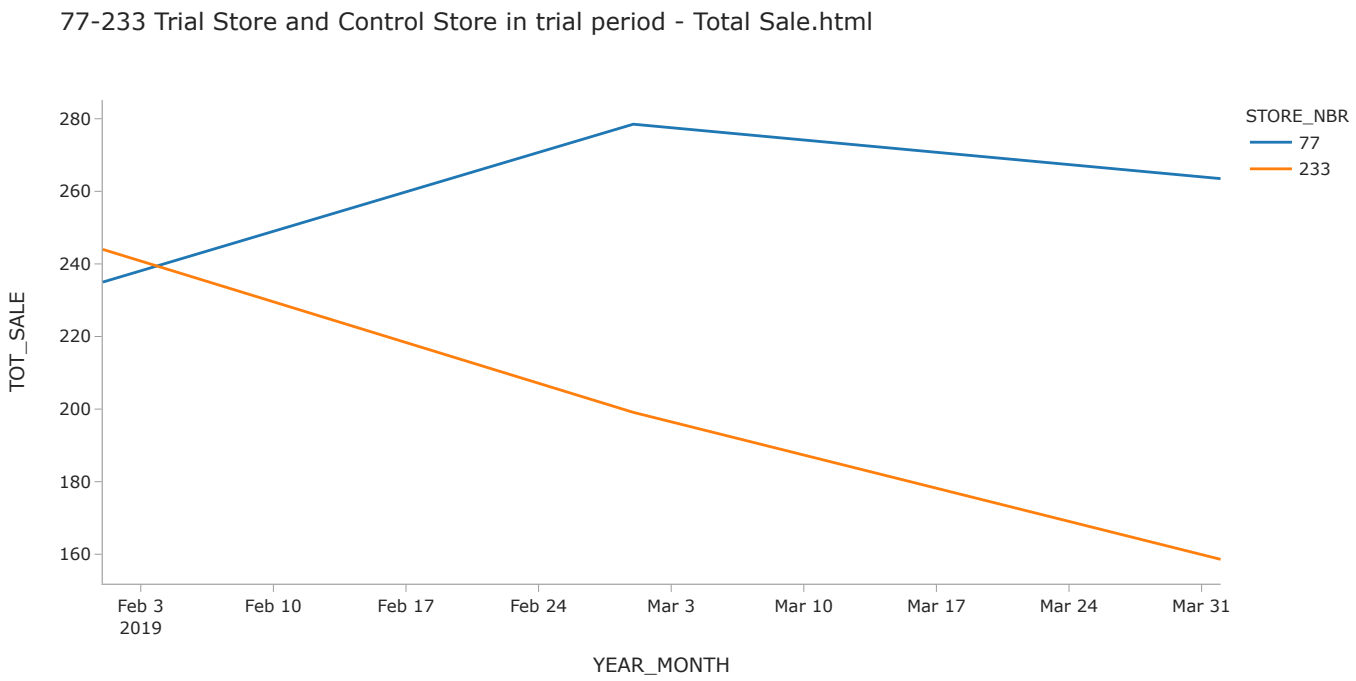
	index	Trial_Str	Control_Str	scoreNSales	scoreNCust	finalControlScore
0	0	77.0	233.0	0.998779	1.000000	0.999389
1	1	86.0	225.0	0.998202	1.000000	0.999101
2	2	88.0	40.0	0.996386	0.992248	0.994317
3	(88, 40)	NaN	NaN	1.000000	1.000000	1.000000

Visual checks on trends based on the drivers for trial period

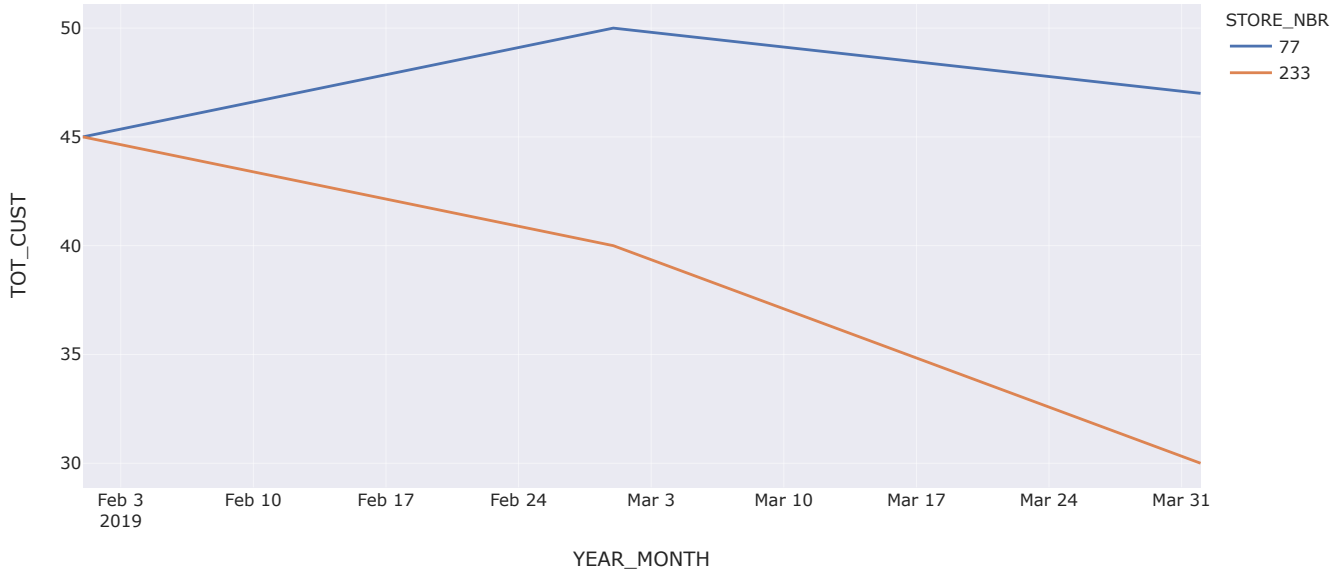
In [24]:

```
trial_control_dict={77:233,86:225,88:40}

for key, val in trial_control_dict.items():
    trial_metrics['YEAR_MONTH']=trial_metrics['YEAR_MONTH'].astype(str)

    matrix=trial_metrics[trial_metrics['STORE_NBR'].isin([key, val])].groupby(['YEAR_MONTH', 'STORE_NBR'])['TOT_SALE','TOT_CUST','TRANS_CUST','PROD_TRANS','AVG_PRICE_UNIT'].sum
().reset_index()

    name1=str(key)+'-'+str(val)+' Trial Store and Control Store in trial period - Total Sale.html'
    fig1 = px.line(matrix, x='YEAR_MONTH',y='TOT_SALE',color='STORE_NBR',template='simple_white',title=name1)
    saveplot(fig1,name1) # saving figure
    fig1.show()

    name2=str(key)+'-'+str(val)+' Trial Store and Control Store in trial period - Total Customer.html'
    fig2 = px.line(matrix, x='YEAR_MONTH',y='TOT_CUST',color='STORE_NBR',template='seaborn',title=name2)
    saveplot(fig2,name2) # saving figure
    fig2.show()

    name3=str(key)+'-'+str(val)+' Trial Store and Control Store - Total Transaction per Customer in trial period.html'
    fig3 = px.line(matrix, x='YEAR_MONTH',y='TRANS_CUST',color='STORE_NBR',template='ggplot2',title=name3)
    saveplot(fig3,name3) # saving figure
    fig3.show()

    name4=str(key)+'-'+str(val)+' Trial Store and Control Store - Total Chips purchase per Transaction in trial period.html'
    fig4 = px.line(matrix, x='YEAR_MONTH',y='PROD_TRANS',color='STORE_NBR',template='presentation',title=name4)
    saveplot(fig4,name4) # saving figure
    fig4.show()

    name5=str(key)+'-'+str(val)+' Trial Store and Control Store - Total Average Price per Unit in trial period.html'
    fig5 = px.line(matrix, x='YEAR_MONTH',y='AVG_PRICE_UNIT',color='STORE_NBR',template='simple_white',title=name5)
    saveplot(fig5,name5) # saving figure
    fig5.show()
```

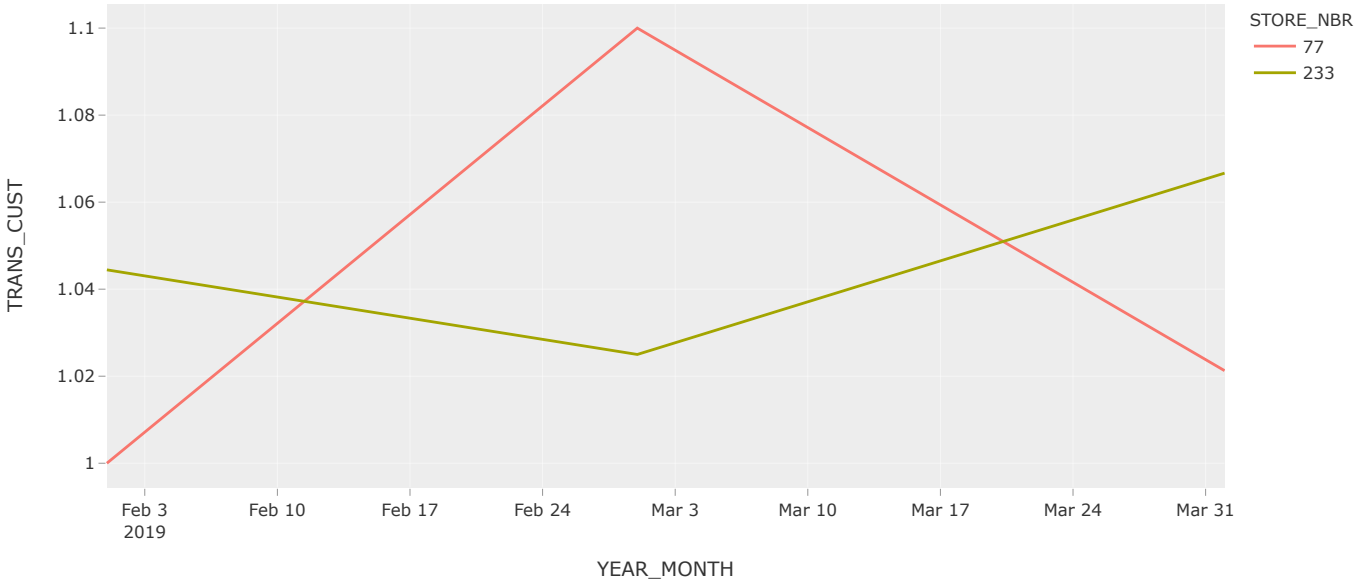
```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:


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

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```



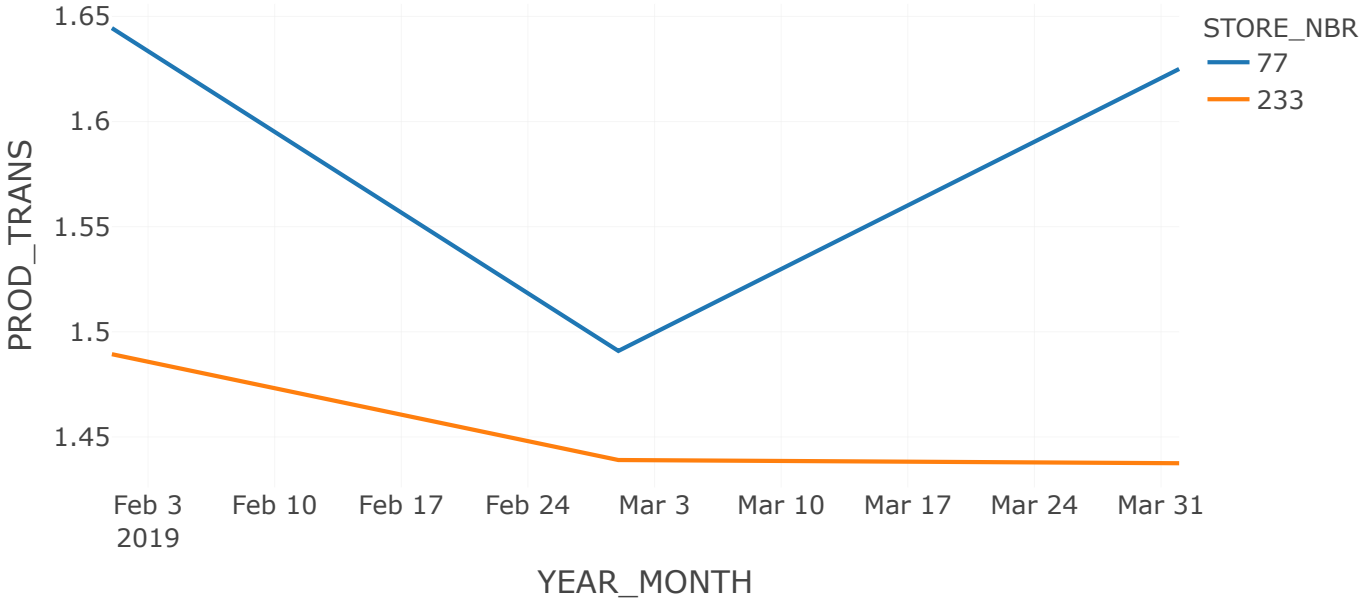
77-233 Trial Store and Control Store in trial period - Total Customer.html



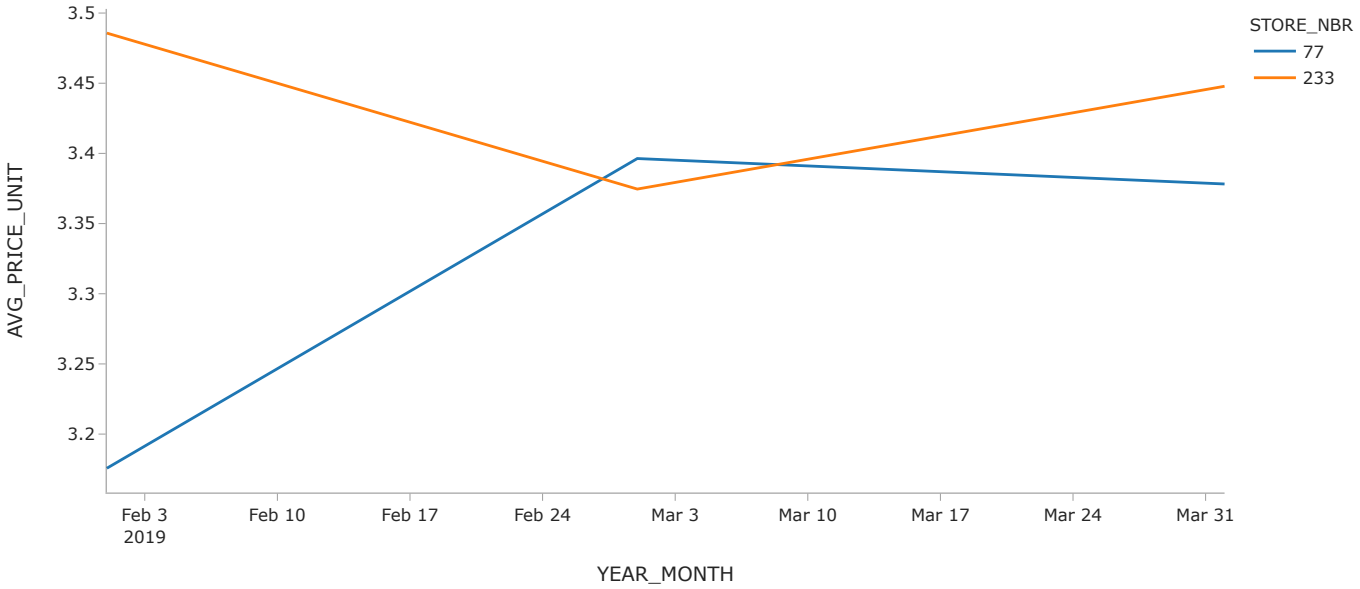
77-233 Trial Store and Control Store - Total Transaction per Customer in trial period.html



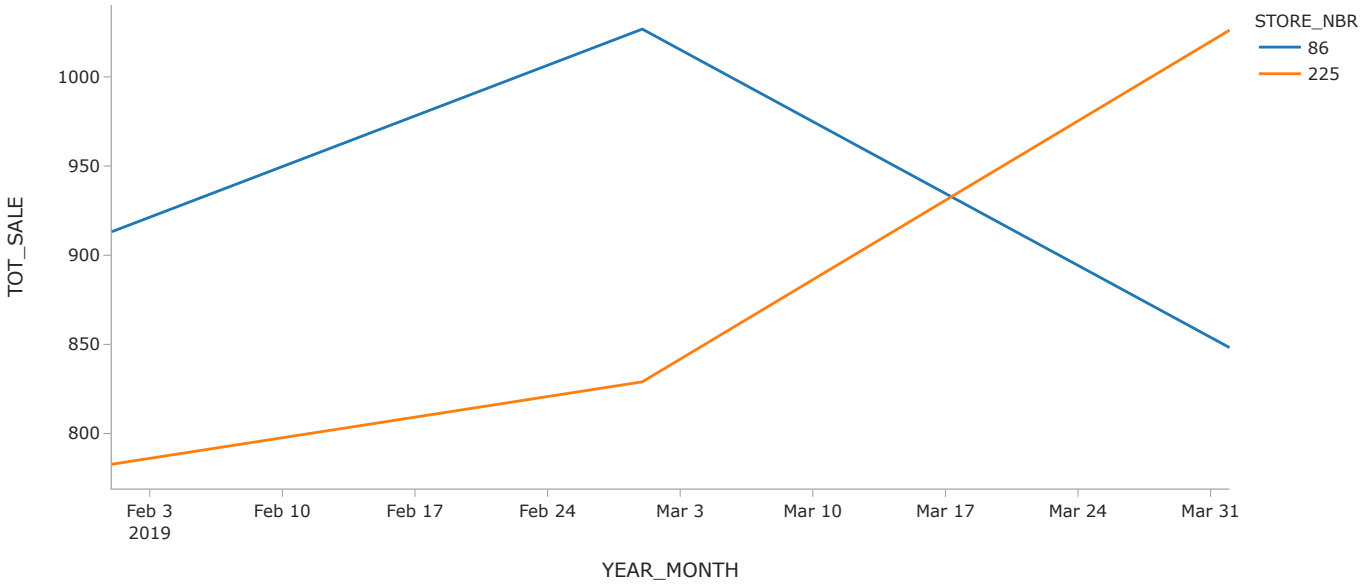
Trial Store and Control Store - Total Chips purchase per Transaction in trial per



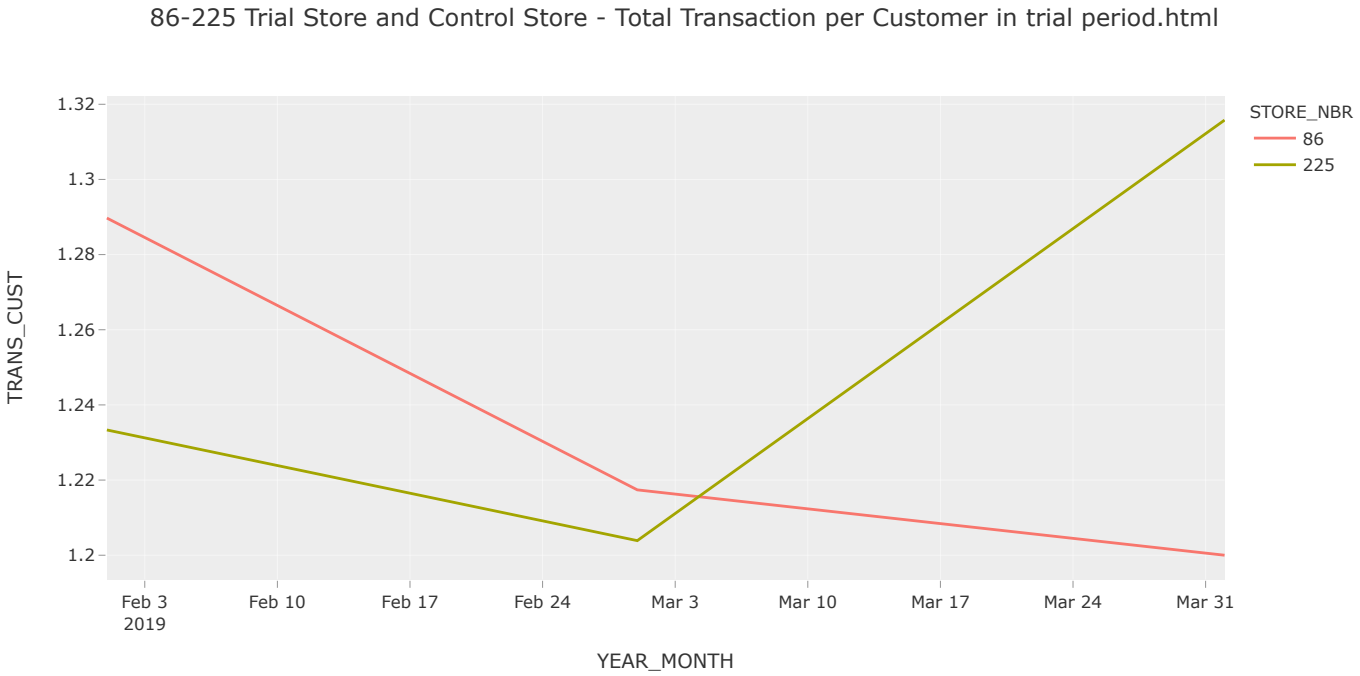
77-233 Trial Store and Control Store - Total Average Price per Unit in trial period.html



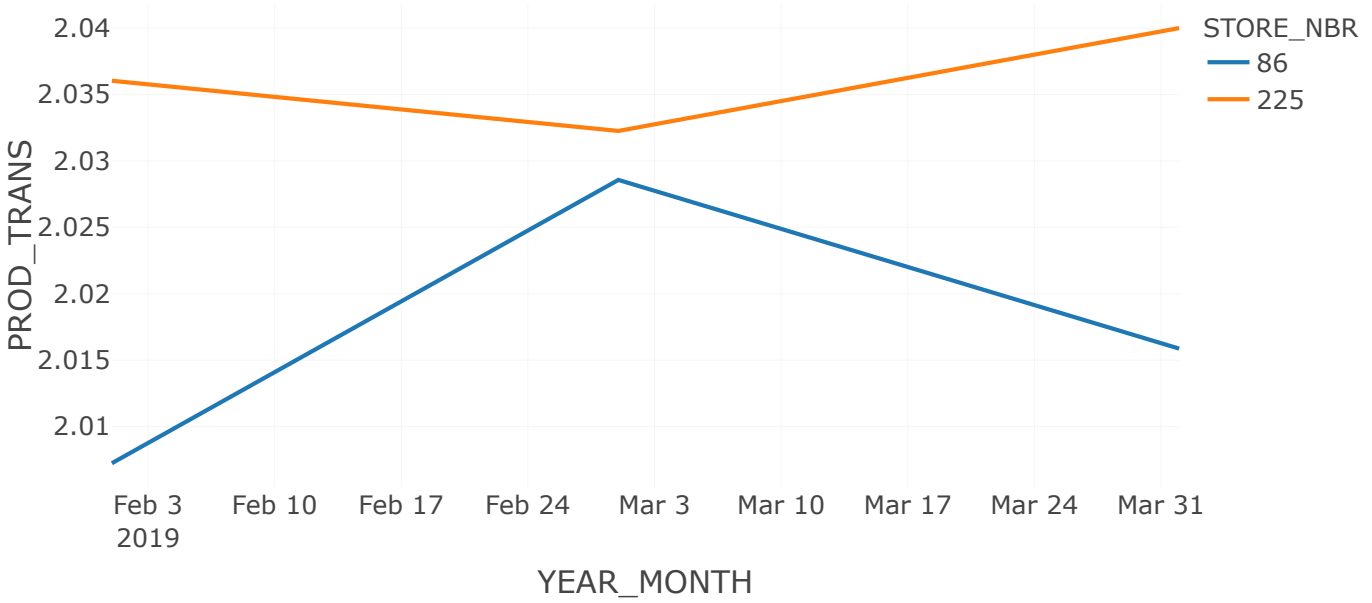
86-225 Trial Store and Control Store in trial period - Total Sale.html



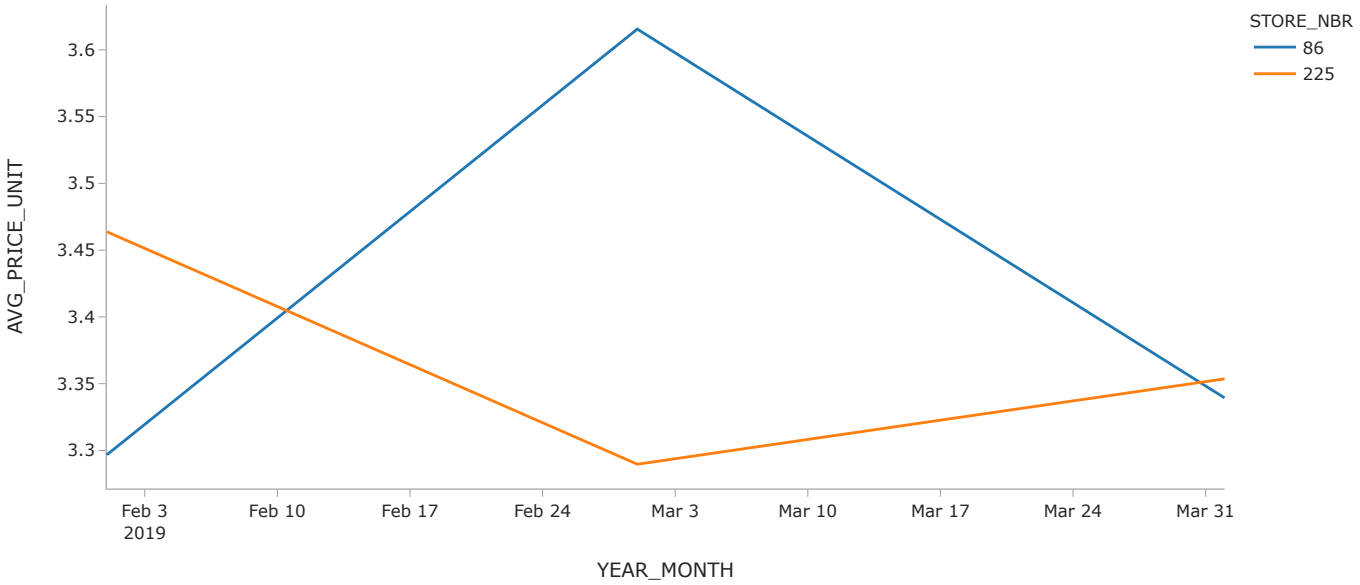

86-225 Trial Store and Control Store in trial period - Total Customer.html
TOT_CUST
115
110
105
100
95
90
Feb 3
2019
Feb 10
Feb 17
Feb 24
Mar 3
Mar 10
Mar 17
Mar 24
Mar 31
YEAR_MONTH
STORE_NBR
86
225


86-225 Trial Store and Control Store - Total Transaction per Customer in trial period.html
TRANS_CUST
1.32
1.3
1.28
1.26
1.24
1.22
1.2
Feb 3
2019
Feb 10
Feb 17
Feb 24
Mar 3
Mar 10
Mar 17
Mar 24
Mar 31
YEAR_MONTH
STORE_NBR
86
225

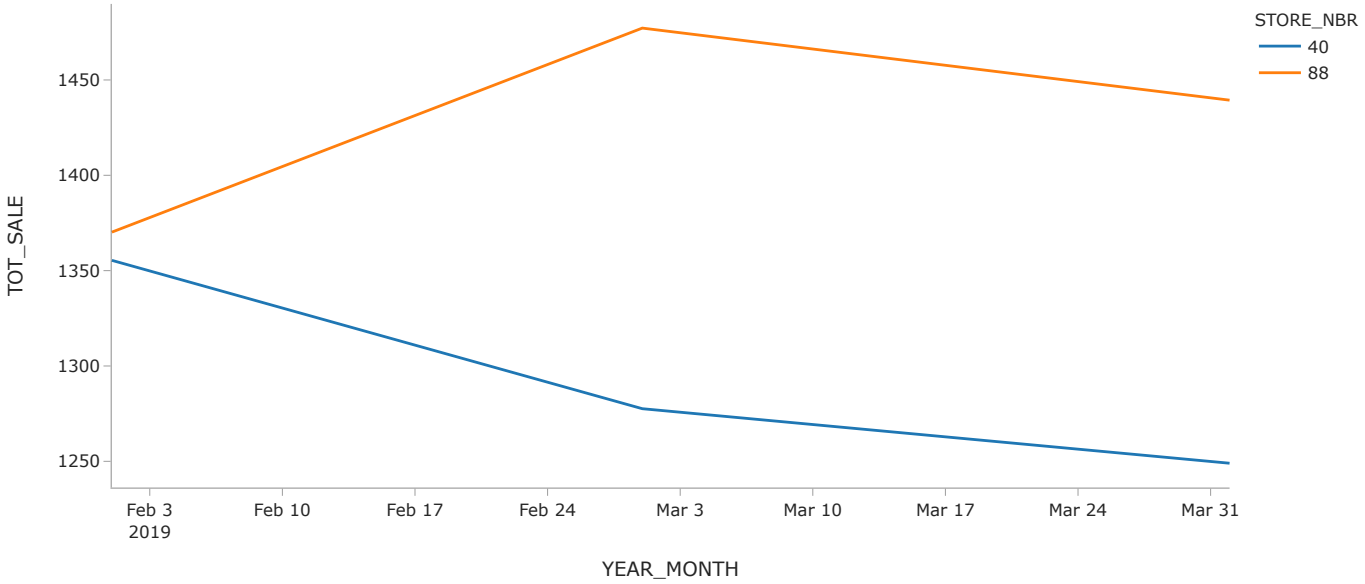
Trial Store and Control Store - Total Chips purchase per Transaction in trial per



86-225 Trial Store and Control Store - Total Average Price per Unit in trial period.html

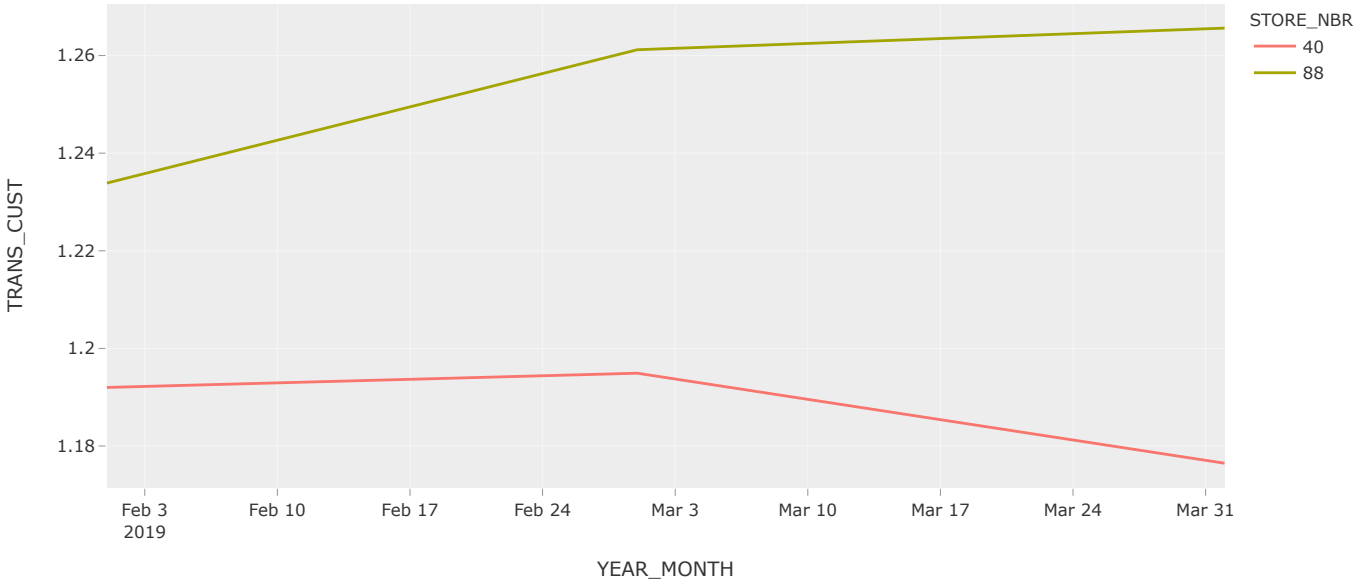


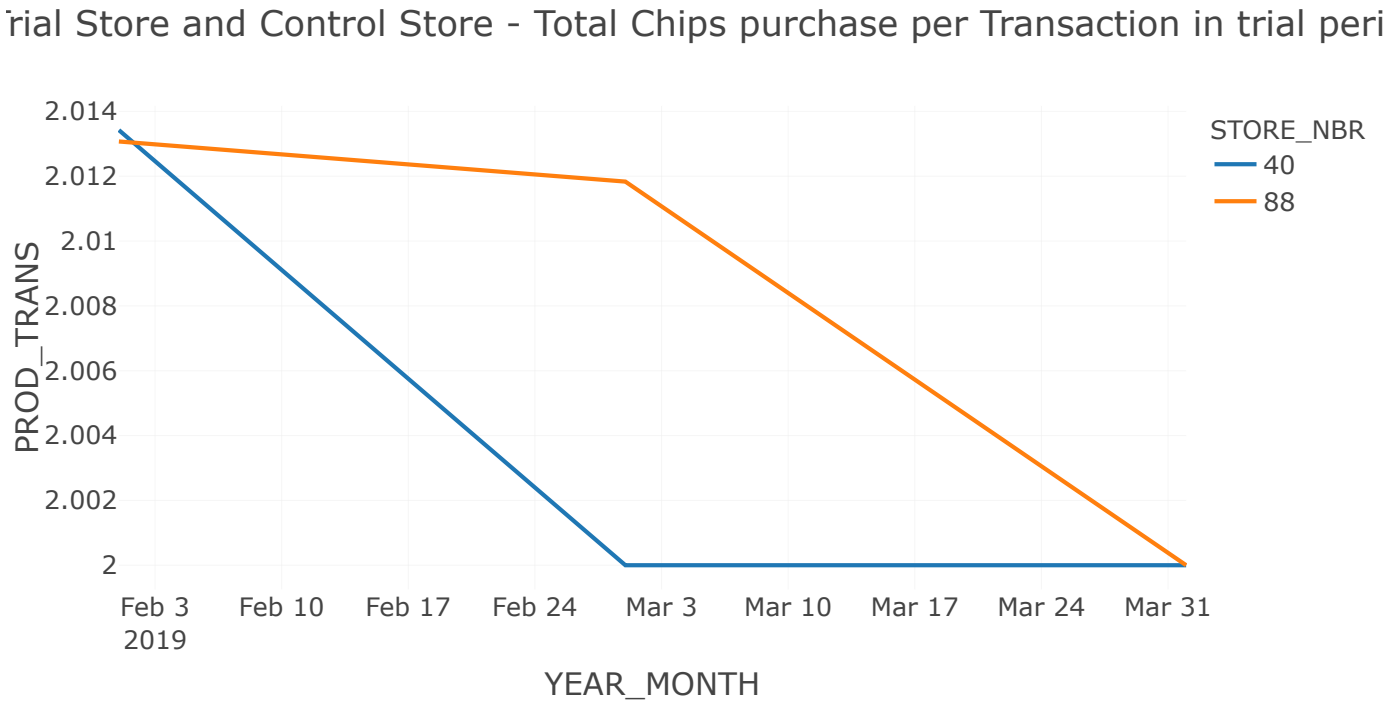
88-40 Trial Store and Control Store in trial period - Total Sale.html



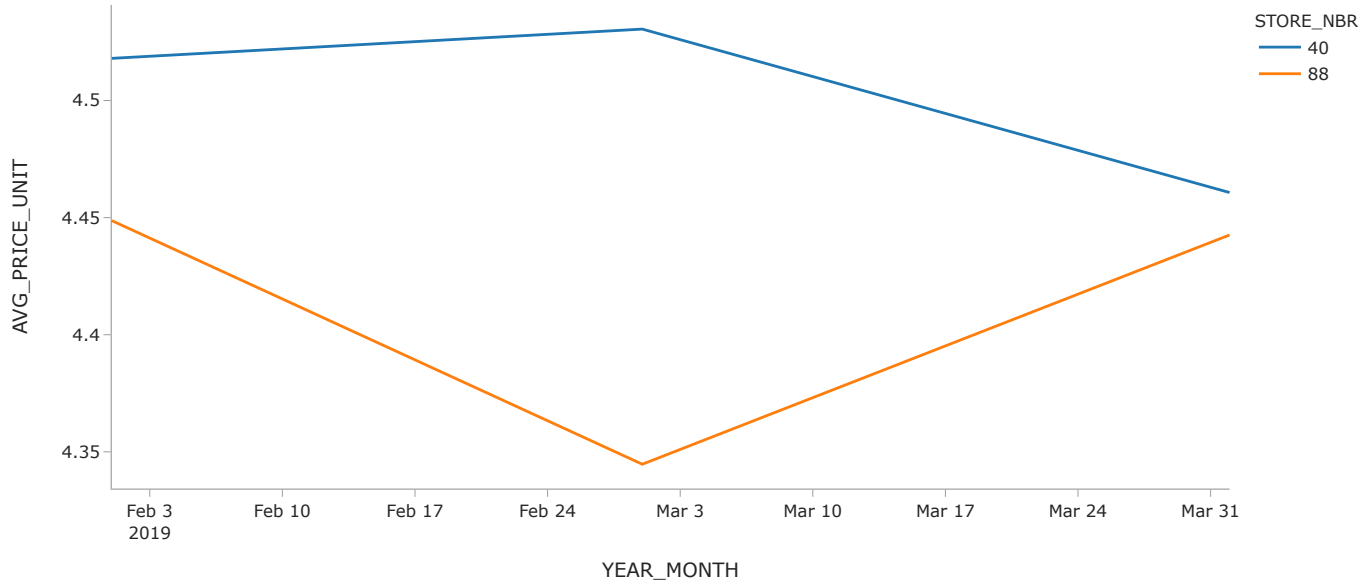
88-40 Trial Store and Control Store in trial period - Total Customer.html

88-40 Trial Store and Control Store - Total Transaction per Customer in trial period.html




rial Store and Control Store - Total Chips purchase per Transaction in trial peri
2.014
2.012
2.01
2.008
2.006
2.004
2.002
2
PROD_TRANS
Feb 3
2019
Feb 10
Feb 17
Feb 24
Mar 3
Mar 10
Mar 17
Mar 24
Mar 31
YEAR_MONTH
STORE_NBR
40
88

88-40 Trial Store and Control Store - Total Average Price per Unit in trial period.html



In [25]:
```
# Creating a function to get store type

def store_type(strnum):

    if strnum in [77,86,88]:
        return 'Trial Store'
    elif strnum in [233,225,40]:
         return 'Control Store'
    else:
        return 'Other store'
```

In [26]:
```
# Getting data for new column 'Store_Type' within 'main dataframe' which categorises store type

type={}
for i in store_metrics.STORE_NBR:
    type[i]=store_type(i)
```

In [27]:
```
type =pd.DataFrame(type.items(),columns=['STORE_NBR','Store_Type'])
type
```

Out[27]:

	STORE_NBR	Store_Type
0	1	Other store
1	2	Other store
2	3	Other store
3	4	Other store
4	5	Other store
...	...	...
255	268	Other store
256	269	Other store
257	270	Other store
258	271	Other store
259	272	Other store

260 rows × 2 columns

In [28]:
```
# Adding column in the metrics dataframe

store_metrics=store_metrics.merge(type,on='STORE_NBR')
store_metrics
```

Out[28]:

	STORE_NBR	YEAR_MONTH	TOT_SALE	TOT_CUST	TRANS_CUST	PROD_TRANS	AVG_PRICE_UNIT	Store_Type
0	1	2018-07	206.9	49	1.061224	1.192308	3.337097	Other store
1	1	2018-08	176.1	42	1.023810	1.255814	3.261111	Other store
2	1	2018-09	278.8	59	1.050847	1.209677	3.717333	Other store
3	1	2018-10	188.1	44	1.022727	1.288889	3.243103	Other store
4	1	2018-11	192.6	46	1.021739	1.212766	3.378947	Other store
...	...	...	...	...	...	...	...	...
3115	272	2019-02	395.5	45	1.066667	1.895833	4.346154	Other store
3116	272	2019-03	442.3	50	1.060000	1.905660	4.379208	Other store
3117	272	2019-04	445.1	54	1.018519	1.909091	4.239048	Other store
3118	272	2019-05	314.6	34	1.176471	1.775000	4.430986	Other store
3119	272	2019-06	312.1	34	1.088235	1.891892	4.458571	Other store

3120 rows × 8 columns

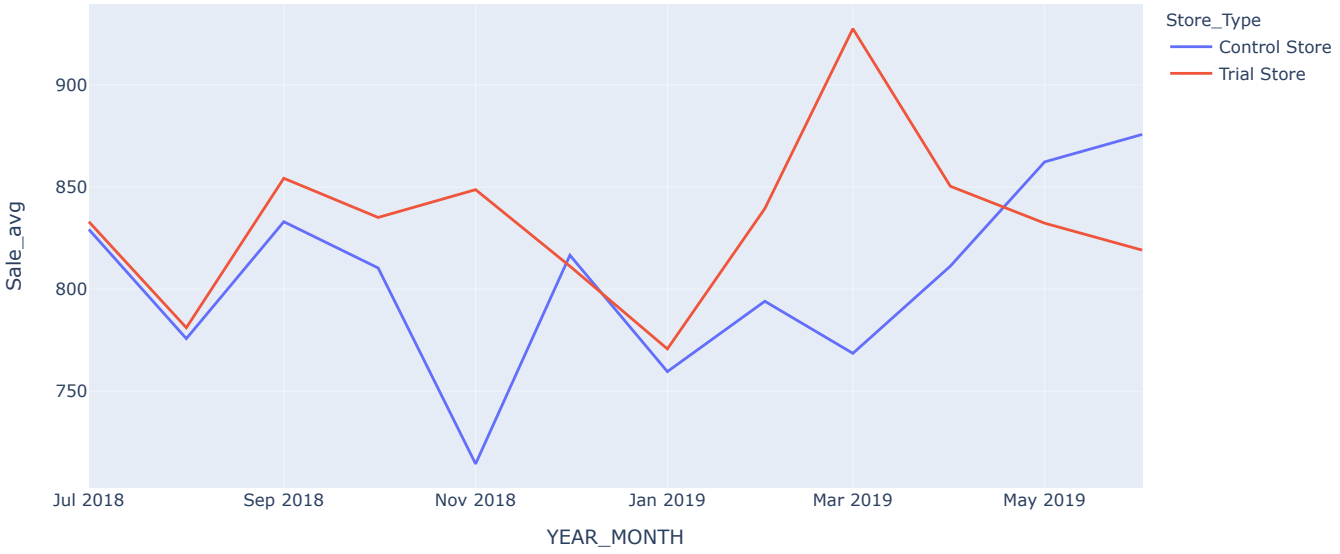
Monthly total sales revenue

In [29]:
```
# Compare stores based on pre-trial total sales average to match trial total sales average

total_sale = store_metrics.groupby(['Store_Type','YEAR_MONTH'])['TOT_SALE'].mean().reset_index(name='Sale_avg')
total_sale = total_sale[~total_sale['Store_Type'].isin(['Other store'])]
total_sale['YEAR_MONTH']=total_sale['YEAR_MONTH'].astype(str)
```

In [30]:
```
name=' Trial Store and Control Store - Total Average Sale comparion - trial and control store'
fig = px.line(total_sale, x='YEAR_MONTH',y='Sale_avg',color='Store_Type',title=name5)
saveplot(fig,name) # saving figure
fig.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\plotly\offline\offline.py:563: UserWarning:

Your filename `Plots/ Trial Store and Control Store - Total Average Sale comparion - trial and control store` didn't end with .html. Adding .html to the end of your file.
```

88-40 Trial Store and Control Store - Total Average Price per Unit in trial period.html



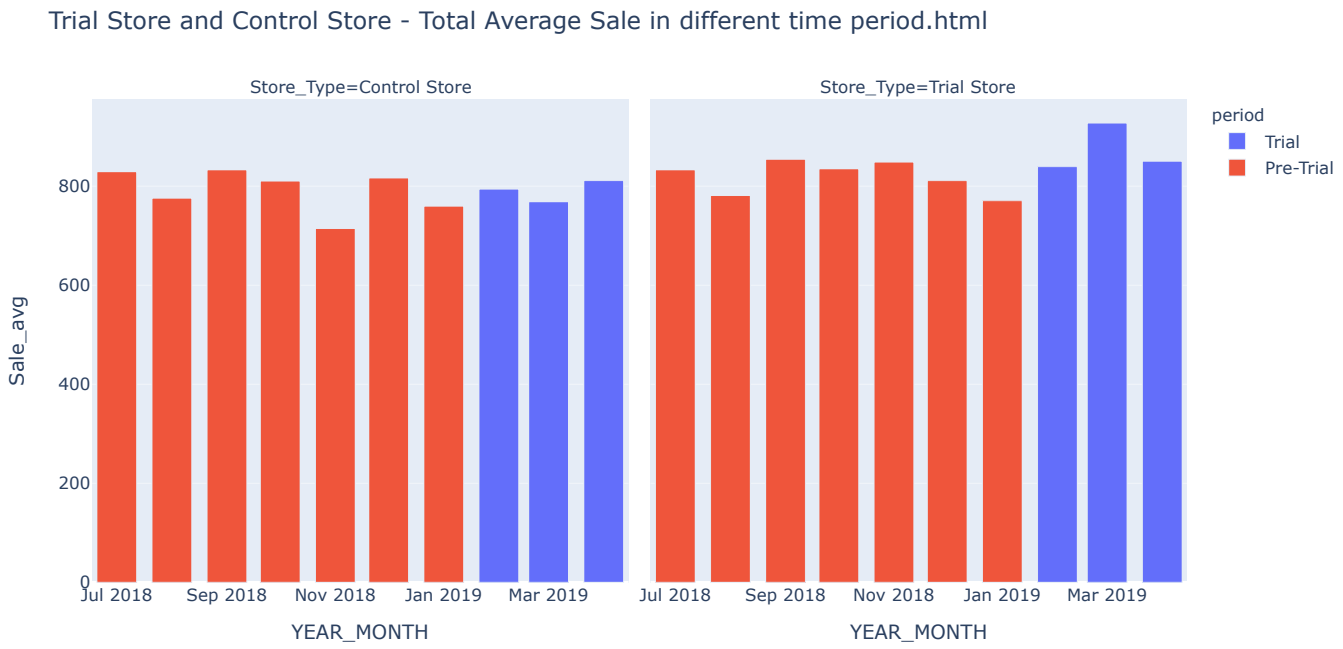
In [39]:
```
# Create a new dataframe to compare total average sale for trial and pre trail period

trial_sale = total_sale.loc[((total_sale['YEAR_MONTH'] > '2019-01')& (total_sale['YEAR_MONTH'] < '2019-05')) ,:]
trial_sale['period']='Trial'
pre_trial_sale = total_sale.loc[total_sale['YEAR_MONTH'] < '2019-02', :]
pre_trial_sale['period']='Pre-Trial'
sale=pd.concat([trial_sale,pre_trial_sale])
sale.head(3)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: 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

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: SettingWithCopyWarning:


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

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

Out[39]:

	Store_Type	YEAR_MONTH	Sale_avg	period
7	Control Store	2019-02	794.066667	Trial
8	Control Store	2019-03	768.566667	Trial
9	Control Store	2019-04	811.266667	Trial

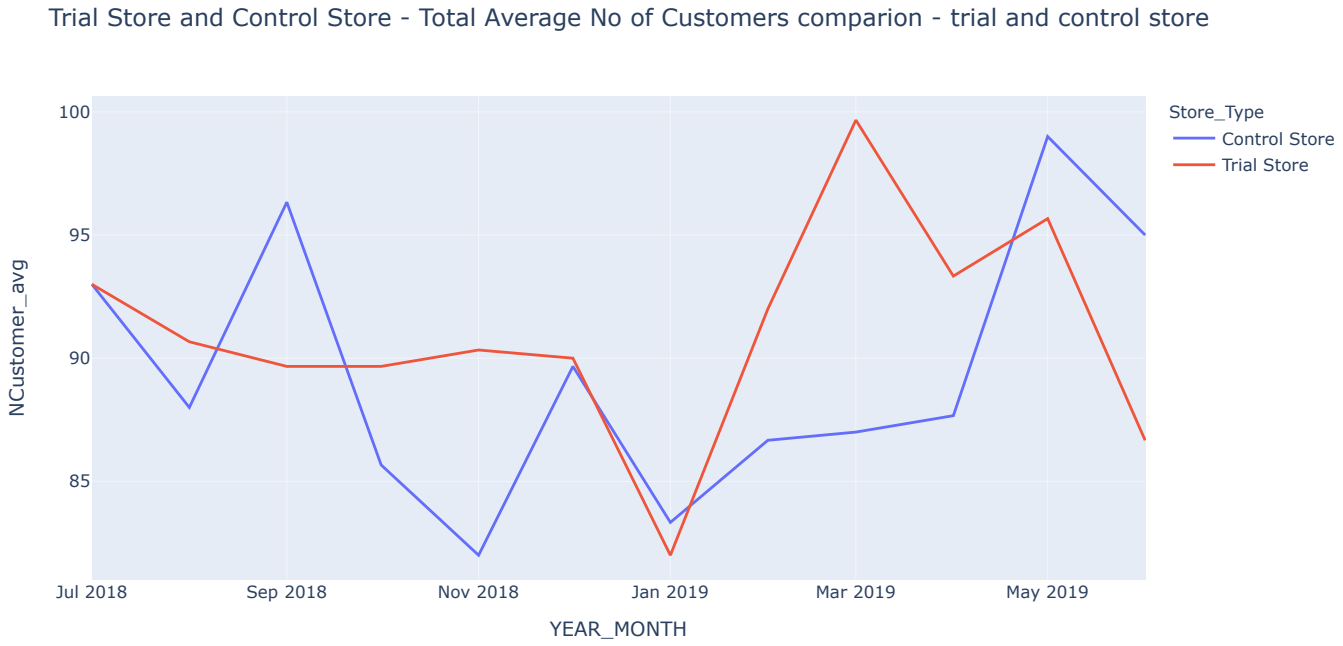
In [51]:
```
name='Trial Store and Control Store - Total Average Sale in different time period.html'
fig = px.bar(sale, x='YEAR_MONTH',y='Sale_avg',color='period',facet_col='Store_Type',title=name)
saveplot(fig,name) # saving figure
fig.show()
```



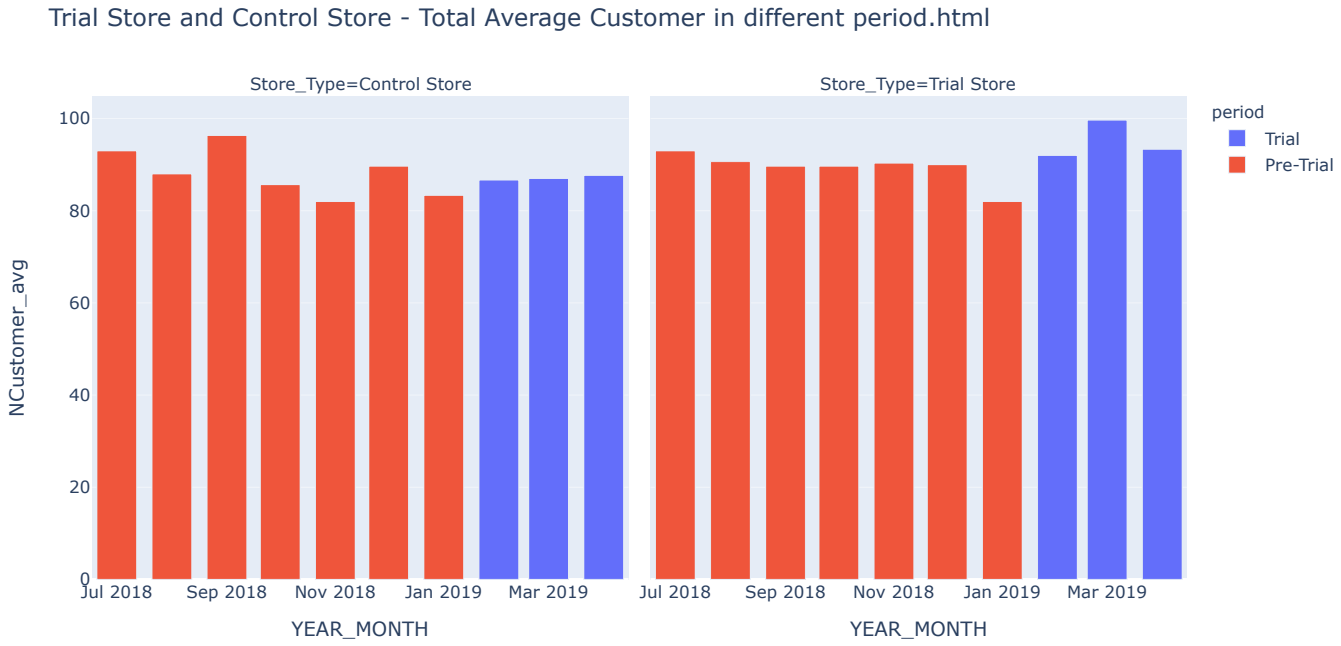
Monthly total number of customers

In [56]:

```
# Compare stores based on pre-trial total sales average to match trial total average no. of customer

total_customer = store_metrics.groupby(['Store_Type','YEAR_MONTH'])['TOT_CUST'].mean().reset_index(name='NCustomer_avg')
total_customer = total_customer[~total_customer['Store_Type'].isin(['Other store'])]
total_customer['YEAR_MONTH']=total_customer['YEAR_MONTH'].astype(str)

name=' Trial Store and Control Store - Total Average No of Customers comparion - trial and control store'
fig = px.line(total_customer, x='YEAR_MONTH',y='NCustomer_avg',color='Store_Type',title=name)
saveplot(fig,name) # saving figure
fig.show()

# Create a new dataframe to compare total average no. of customers for trial and pre trail period

trial_sale = total_customer.loc[((total_customer['YEAR_MONTH'] > '2019-01')& (total_customer['YEAR_MONTH'] < '2019-05')) ,:]
trial_sale['period']='Trial'
pre_trial_sale = total_customer.loc[total_customer['YEAR_MONTH'] < '2019-02', :]
pre_trial_sale['period']='Pre-Trial'
sale=pd.concat([trial_sale,pre_trial_sale])
sale.head(3)

name='Trial Store and Control Store - Total Average Customer in different period.html'
fig = px.bar(sale, x='YEAR_MONTH',y='NCustomer_avg',color='period',facet_col='Store_Type',title=name)
saveplot(fig,name) # saving figure
fig.show()
```




```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: 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

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:17: 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
```

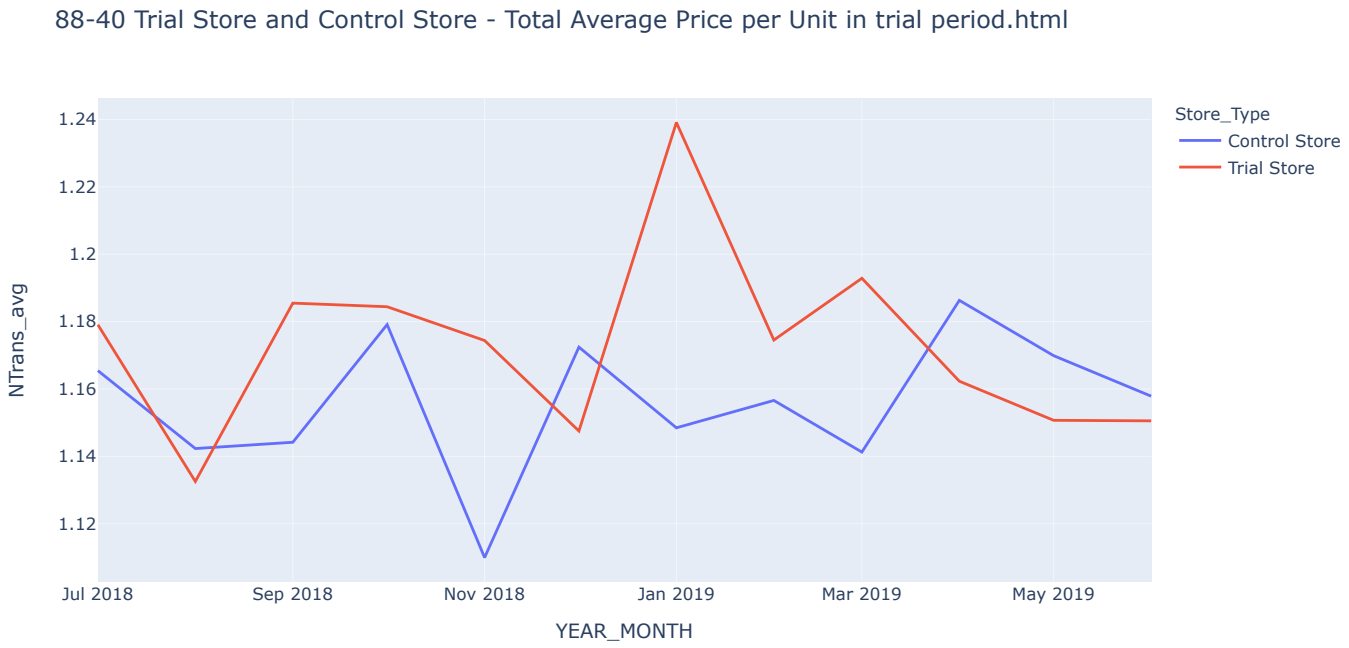


Monthly average number of transactions per customer

In [54]:
```
# Compare stores based on pre-trial total sales average to match trial total average transaction per customer

total_customer = store_metrics.groupby(['Store_Type','YEAR_MONTH'])['TRANS_CUST'].mean().reset_index(name='NTrans_avg')
total_customer = total_customer[~total_customer['Store_Type'].isin(['Other store'])]
total_customer['YEAR_MONTH']=total_customer['YEAR_MONTH'].astype(str)

name=' Trial Store and Control Store - Total Average No of Transaction per customer comparion - trial and control store'
fig = px.line(total_customer, x='YEAR_MONTH',y='NTrans_avg',color='Store_Type',title=name5)
saveplot(fig,name) # saving figure
fig.show()

# Create a new dataframe to compare total average transaction per customer for trial and pre trail period

trial_sale = total_customer.loc[((total_customer['YEAR_MONTH'] > '2019-01')& (total_customer['YEAR_MONTH'] < '2019-05')) ,:]
trial_sale['period']='Trial'
pre_trial_sale = total_customer.loc[total_customer['YEAR_MONTH'] < '2019-02', :]
pre_trial_sale['period']='Pre-Trial'
sale=pd.concat([trial_sale,pre_trial_sale])
sale.head(3)

name='Trial Store and Control Store - Total Average Transaction per customer in different period.html'
fig = px.bar(sale, x='YEAR_MONTH',y='NTrans_avg',color='period',facet_col='Store_Type', title=name)
saveplot(fig,name) # saving figure
fig.show()
```

C:\ProgramData\Anaconda3\lib\site-packages\plotly\offline\offline.py:563: UserWarning:

Your filename `Plots/ Trial Store and Control Store - Total Average No of Transaction per customer comparion - trial and control store` didn't end with .html. Adding .html to the end of your file.



C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: SettingWithCopyWarning:

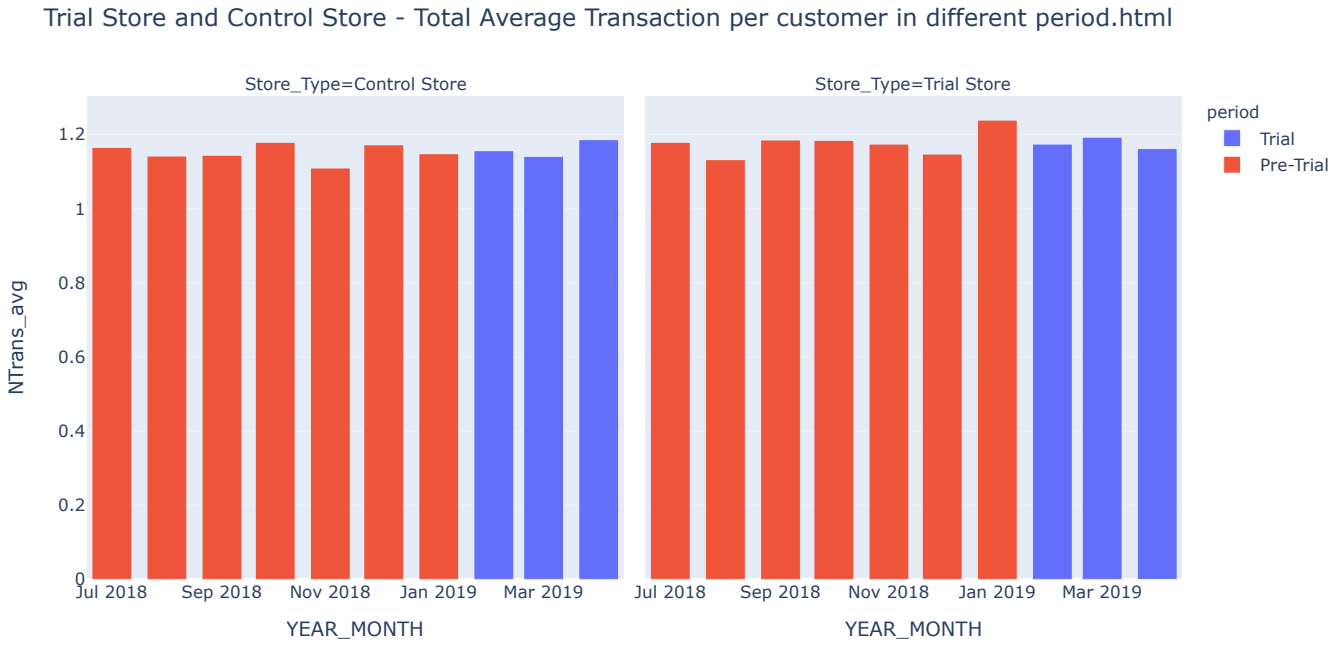
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

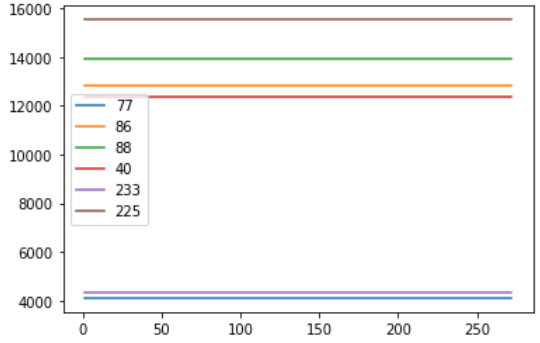
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:17: SettingWithCopyWarning:

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



In [60]:
```
store_matrix=store_metrics.groupby('STORE_NBR')['TOT_SALE'].sum().reset_index(name='Total_sale')

trial_control=['77','86','88','40','233','225']

for store in trial_control:
    sns.lineplot(data=store_matrix.loc[int(store)],y='Total_sale',x=store_matrix['STORE_NBR'].unique(),label=store)
```



Conclusion:

It looks like the number of sales is significantly higher in all of the three trial period months. This seems to suggest that the trial had a significant impact on increasing the number of transactions in trial store 88 but as we saw, customer numbers were not significantly higher. We should check with the Category Manager if there were special deals in the trial store that were may have resulted in promotion and can attract more customer to come, impacting the results.

In [ ]: