Quantium Virtual Internsdhip Task 2

Experimentation and uplift testing

Task -

1. Consider the monthly sales experience of each store (Total Sales, Total Number of Customer, Transaction per customer)
2. Create a measure to compare different control stores to each of the trial stores (Pearson's Correlation , Manhattan Distance)
3. Compare each trial and control pair during the trial period

In [1]:
```
#Import Necessary Library/Module
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#Set visualization theme and read data file
sns.set_theme(palette="pastel")
data = pd.read_csv("QVI_data.csv")
```

In [3]:
```
sns.set_theme()
```

Taking at look at the data before working on it.

In [4]:
```
data.shape
```

Out[4]:
```
(264834, 12)
```

In [5]:
```
data.nunique()
```

Out[5]:
```
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
```

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264834 entries, 0 to 264833
Data columns (total 12 columns):
LYLTY_CARD_NBR      264834 non-null int64
DATE                264834 non-null object
STORE_NBR           264834 non-null int64
TXN_ID              264834 non-null int64
PROD_NBR            264834 non-null int64
PROD_NAME           264834 non-null object
PROD_QTY            264834 non-null int64
TOT_SALES           264834 non-null float64
PACK_SIZE           264834 non-null int64
BRAND               264834 non-null object
LIFESTAGE           264834 non-null object
PREMIUM_CUSTOMER    264834 non-null object
dtypes: float64(1), int64(6), object(5)
memory usage: 19.2+ MB
```

In [7]:
```
data.sample(5)
```

Out[7]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND
258237	265218	2019-04-19	265	263557	74	Tostitos Splash Of Lime 175g	1	4.4	175	TOSTITOS

63812	67225	2018-12-27	67	65159	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	2	9.2	150	KETTLE
184453	185392	2019-03-20	185	188271	16	Smiths Crinkle Chips Salt & Vinegar 330g	1	5.7	330	SMITHS
94029	96193	2019-02-24	96	96013	16	Smiths Crinkle Chips Salt & Vinegar 330g	2	11.4	330	SMITHS
149383	153136	2018-08-26	153	152482	109	Pringles Barbeque 134g	2	7.4	134	PRINGLES

TRIAL PERIOD FEB2019-APR2019

- Trial Stores = 77 , 86 , 88
- Control Store must have sales every month from July2018 - June2019
- Find store with similar MONTHLY OVERALL SALES , MONTHLY NUMBER OF CUSTOMER & NUMBER OF TRANSACTION PER CUSTOMER.

The Code below creates a new Month column and removes any data/store that does not have 12months worth of sales.

In [8]:
```
#Create Month column
data["DATE"] = pd.to_datetime(data["DATE"],dayfirst=True)
data["MONTH"] = data["DATE"].dt.to_period("m")

# Find for store with 12 months of sales
months = (data.groupby(["STORE_NBR"])["MONTH"].nunique() == 12)

#Dropping stores with less than 12 months of sales
data = data.loc[data["STORE_NBR"].isin(months[months].index)]
```

After creating a new Month column in the main dataset & removing unwanted stores, i create a new dataset that contains the Sum of product sales and unique customer transaction for each Store by month.

In [9]:
```
#Group Data by Store Number & Month.

#Sum of Total Sales & Product Quantity-
sales = data.groupby(["STORE_NBR","MONTH"])["TOT_SALES","PROD_QTY"].sum().reset_index()
#Count Unique Customer & Transaction-
lylty = data.groupby(["STORE_NBR","MONTH"])["LYLTY_CARD_NBR","TXN_ID"].nunique().reset_index()

#Merge Data above to create a single DataFrame.
by_month = pd.merge(sales,lylty,on=["STORE_NBR","MONTH"])
by_month.sample(5)
```

Out[9]:

	STORE_NBR	MONTH	TOT_SALES	PROD_QTY	LYLTY_CARD_NBR	TXN_ID
929	82	2018-12	306.5	85	42	43
341	30	2018-12	1014.0	266	99	131
428	38	2019-03	225.1	66	39	43
498	45	2019-01	957.4	278	107	137
2356	205	2018-11	224.2	59	36	37

With each store grouped by its monthly data. We can now compare each Trial Store against other store to find for matching Control Store.
We will be looking for stores that have equal correlation and differences in Sales & Customer.

In [10]:
```
#Created 2 seperate dataframe one for before trial and one during.
#I do this to write less code. Not efficient for space.
before_trial = by_month.loc[by_month["MONTH"] < "2019-02"]
during_trial = by_month.loc[(by_month["MONTH"] >= "2019-02") & (by_month["MONTH"] <= "2019-04")]
```

In [11]:
```
#This function finds for similar store.
#This fucntion finds for store similar using both Pearsons Correlation and Sum of Absolute Difference.
#stores = dataframe , t_num = trail store number , by_col = column to compare , corr_weight = correlati
on weight(0-1)
#Increase corr_weight to prioritize correlation over difference. Best at (0.5)
def find_similar(stores,t_num,by_col,corr_weight):
    #Variable for trial store and not trial store.
    trial = stores.loc[stores["STORE_NBR"] == t_num]
    store = stores.loc[stores["STORE_NBR"] != t_num]
    #Compare each store against Trial store and create new column for correlation number and distance/d
ifference number.
    for i in store["STORE_NBR"].unique():
        store.loc[store["STORE_NBR"] == i , "corrcoef"] =  np.corrcoef(trial[by_col].values,store.loc[s
tore["STORE_NBR"] == i , by_col].values)[0,1]
        store.loc[store["STORE_NBR"] == i , "distance"] = sum(abs(trial[by_col].values - store.loc[stor
e["STORE_NBR"] == i , by_col].values))
    #mag_dist = Magnitude of Distance / Percentage of Difference.
    store.loc[:,"mag_dist"] = 1 - ((store["distance"] - store["distance"].min()) / (  store["distance"]
.max() -  store["distance"].min() ))
    #best = store with best correlation and differences.
    store.loc[:,"best"] = (store["corrcoef"] * corr_weight) + ( store["mag_dist"] * (1 - corr_weight) )
    return store
```

In [12]:
```
#Return Store with most similar in sales and customer in terms of correlation and differences.
#left = sales/customer data , right = sales/customer data , on = on which column (similar to merge 'o
n') , cols = Columns needed.
#Can only be use if data is from find_similar().
def final(left,right,ons,cols):
    df = pd.merge(left[cols],right[cols],on=ons)
    df.loc[:,"final"] = (df["best_x"] * .5) + (df["best_y"] * .5)
    return df.loc[df["final"] == df["final"].max(),"STORE_NBR"].unique()[0]
```

In [13]:
```
#Find similar in Sales & Customer before trial period using find_similar()
sales_77 = find_similar(before_trial,77,"TOT_SALES",.5)
lylty_77= find_similar(before_trial,77,"LYLTY_CARD_NBR",.5)

sales_86 = find_similar(before_trial,86,"TOT_SALES",.5)
lylty_86= find_similar(before_trial,86,"LYLTY_CARD_NBR",.5)

sales_88 = find_similar(before_trial,88,"TOT_SALES",.5)
lylty_88= find_similar(before_trial,88,"LYLTY_CARD_NBR",.5)
```

```
c:\users\user\appdata\local\programs\python\python37-32\lib\site-packages\pandas\core\indexing.py:37
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: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  self.obj[key] = _infer_fill_value(value)
c:\users\user\appdata\local\programs\python\python37-32\lib\site-packages\pandas\core\indexing.py:49
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: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  self.obj[item] = s
```

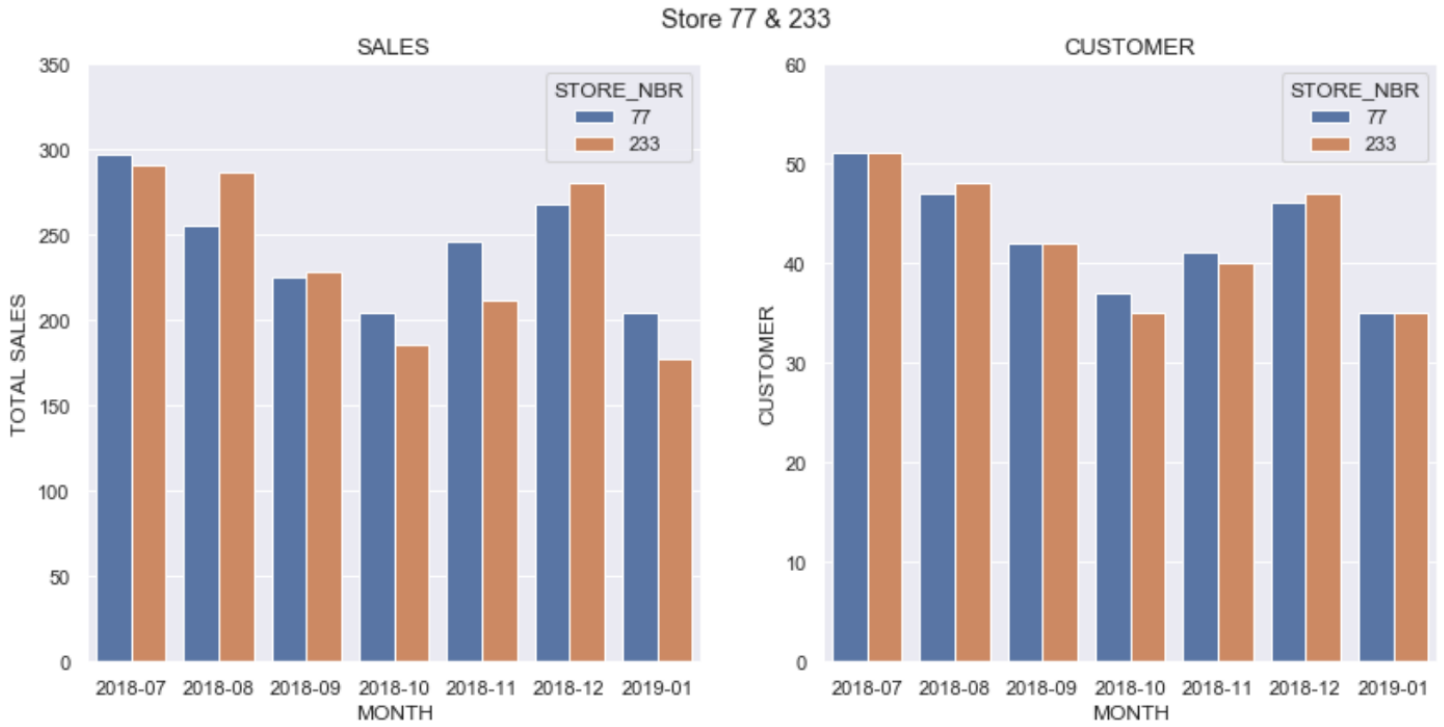
In [14]:
```
#Merge and look for control store that is most similar to trial store in terms of sales and customer.
control_77 = final(sales_77,lylty_77,["STORE_NBR","MONTH"],["STORE_NBR","MONTH","best"])
control_86 = final(sales_86,lylty_86,["STORE_NBR","MONTH"],["STORE_NBR","MONTH","best"])
control_88 = final(sales_88,lylty_88,["STORE_NBR","MONTH"],["STORE_NBR","MONTH","best"])
print("Trial Store with selected Control Store-")
print(f"{77}={control_77} , {86}={control_86} , {88}={control_88} ")
```

```
Trial Store with selected Control Store-
77=233 , 86=155 , 88=237
```

Now that we have our control store for each trial store we can plot them to see if they are similar.

In [15]:
```
fig , axes = plt.subplots(1,2,figsize=(12,6))

fig.tight_layout(pad=3)
sns.barplot(x="MONTH",y="TOT_SALES",hue="STORE_NBR",data=before_trial.loc[before_trial["STORE_NBR"].isi
n([77,control_77])],ax=axes[0])
sns.barplot(x="MONTH",y="LYLTY_CARD_NBR",hue="STORE_NBR",data=before_trial.loc[before_trial["STORE_NBR"
].isin([77,control_77])],ax=axes[1])
axes[0].set_ylim(0,350)
axes[0].set_ylabel("TOTAL SALES",fontsize = 12.0)
axes[0].set_xlabel("MONTH",fontsize=12.0)
axes[0].set_title("SALES",fontsize = 13.0)
axes[1].set_ylim(0,60)
axes[1].set_ylabel("CUSTOMER",fontsize = 12.0)
axes[1].set_xlabel("MONTH",fontsize=12.0)
axes[1].set_title("CUSTOMER",fontsize = 13.0)
fig.suptitle("Store 77 & 233")
plt.show()
```

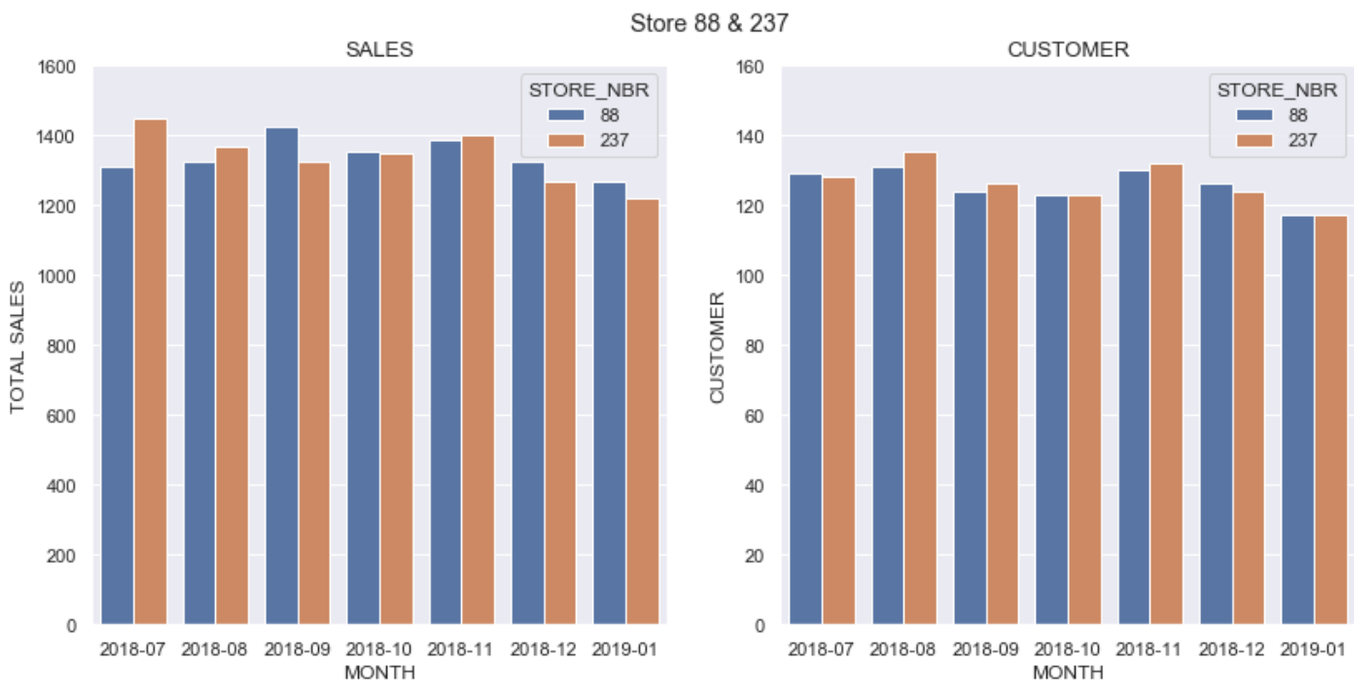


In [16]:
```
fig , axes = plt.subplots(1,2,figsize=(12,6))

fig.tight_layout(pad=3)
sns.barplot(x="MONTH",y="TOT_SALES",hue="STORE_NBR",data=before_trial.loc[before_trial["STORE_NBR"].isi
n([86,control_86])],ax=axes[0])
sns.barplot(x="MONTH",y="LYLTY_CARD_NBR",hue="STORE_NBR",data=before_trial.loc[before_trial["STORE_NBR"
].isin([86,control_86])],ax=axes[1])
axes[0].set_ylim(0,1200)
axes[0].set_ylabel("TOTAL SALES",fontsize = 12.0)
axes[0].set_xlabel("MONTH",fontsize=12.0)
axes[0].set_title("SALES",fontsize = 13.0)
axes[1].set_ylim(0,120)
axes[1].set_ylabel("CUSTOMER",fontsize = 12.0)
axes[1].set_xlabel("MONTH",fontsize=12.0)
axes[1].set_title("CUSTOMER",fontsize = 13.0)
fig.suptitle("Store 86 & 155")
plt.show()
```

In [17]:
```
fig , axes = plt.subplots(1,2,figsize=(12,6))

fig.tight_layout(pad=3)
sns.barplot(x="MONTH",y="TOT_SALES",hue="STORE_NBR",data=before_trial.loc[before_trial["STORE_NBR"].isi
n([88,control_88])],ax=axes[0])
sns.barplot(x="MONTH",y="LYLTY_CARD_NBR",hue="STORE_NBR",data=before_trial.loc[before_trial["STORE_NBR"
].isin([88,control_88])],ax=axes[1])
axes[0].set_ylim(0,1600)
axes[0].set_ylabel("TOTAL SALES",fontsize = 12.0)
axes[0].set_xlabel("MONTH",fontsize=12.0)
axes[0].set_title("SALES",fontsize = 13.0)
axes[1].set_ylim(0,160)
axes[1].set_ylabel("CUSTOMER",fontsize = 12.0)
axes[1].set_xlabel("MONTH",fontsize=12.0)
axes[1].set_title("CUSTOMER",fontsize = 13.0)
fig.suptitle("Store 88 & 237")
plt.show()
```



Now that we have found our control store for each trial store. We can perform our hypothesis testing to see if there is any improvement to our trial store during the Trial Period(Feb2019 - Apr2019).
Our null hypothesis should be that there is no diffrence between sales and customer before and during trial period.

- Scale Control Store to match trial store before trial period.
- Find the Percentage of Difference between Trial Store and Control Store.
- We will use the Percentage of Difference before trial period to be our Null Hypothesis Value.

In [18]:
```
#This variable contains only trial store and control store during trial period.
during_trial = during_trial.loc[during_trial["STORE_NBR"].isin([control_77,control_86,control_88,77,86,
88])]
```

In [19]:
```
#Convert Period to Datetime for plotting
before_trial.loc[:,"MONTH"] = before_trial["MONTH"].dt.to_timestamp()
during_trial.loc[:,"MONTH"] = during_trial["MONTH"].dt.to_timestamp()
```

```
c:\users\user\appdata\local\programs\python\python37-32\lib\site-packages\pandas\core\indexing.py:49
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: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  self.obj[item] = s
```

In [20]:
```
#This function create a new column to store scaled sales and customer data.
#The scaling factor will base on the first dataset in datasets. datasets[0]
def transform_scale(datasets , trial, control , column, new_column):
    scale = datasets[0].loc[datasets[0]["STORE_NBR"] == trial,column].sum() / datasets[0].loc[datasets[
0]["STORE_NBR"] == control,column].sum()
    print("Scaled By : ",scale)
    for i in datasets:
        i.loc[i["STORE_NBR"] == control , new_column] = i.loc[ i["STORE_NBR"] == control,column] * scal
e
```

In [21]:
```
#Scaling control store sales value
transform_scale([before_trial,during_trial],77,control_77,"TOT_SALES","SCALED_SALES")
transform_scale([before_trial,during_trial],86,control_86,"TOT_SALES","SCALED_SALES")
transform_scale([before_trial,during_trial],88,control_88,"TOT_SALES","SCALED_SALES")
during_trial.sample(5)
```

```
Scaled By :  1.023617303289553
Scaled By :  0.9700651481287746
Scaled By :  1.0015583306649594
```

```
c:\users\user\appdata\local\programs\python\python37-32\lib\site-packages\pandas\core\indexing.py:37
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: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  self.obj[key] = _infer_fill_value(value)
```

Out[21]:

	STORE_NBR	MONTH	TOT_SALES	PROD_QTY	LYLTY_CARD_NBR	TXN_ID	SCALED_SALES
2660	233	2019-03-01	199.1	59	40	41	203.802205
871	77	2019-02-01	235.0	74	45	45	NaN
873	77	2019-04-01	263.5	78	47	48	NaN
1772	155	2019-03-01	804.4	240	94	118	780.320405
2661	233	2019-04-01	158.6	46	30	32	162.345704

In [22]:
```
#Scaling control store customer value
transform_scale([before_trial,during_trial],77,control_77,"LYLTY_CARD_NBR","SCALED_CUST")
transform_scale([before_trial,during_trial],86,control_86,"LYLTY_CARD_NBR","SCALED_CUST")
transform_scale([before_trial,during_trial],88,control_88,"LYLTY_CARD_NBR","SCALED_CUST")
```

```
Scaled By :  1.0033557046979866
Scaled By :  1.0
Scaled By :  0.9943502824858758
```

In [23]:
```
before_trial.loc[before_trial["STORE_NBR"] == control_77]
```

Out[23]:

	STORE_NBR	MONTH	TOT_SALES	PROD_QTY	LYLTY_CARD_NBR	TXN_ID	SCALED_SALES	SCALED_CUST
2652	233	2018-07-01	290.7	88	51	54	297.565550	51.171141
2653	233	2018-08-01	285.9	80	48	50	292.652187	48.161074
2654	233	2018-09-01	228.6	70	42	45	233.998916	42.140940
2655	233	2018-10-01	185.7	56	35	36	190.085733	35.117450
2656	233	2018-11-01	211.6	62	40	41	216.597421	40.134228
2657	233	2018-12-01	279.8	75	47	50	286.408121	47.157718
2658	233	2019-01-01	177.5	47	35	35	181.692071	35.117450

There are 7 month in pre-trial period. We will be using 7-1 Degrees of Freedom & a Alpha of 0.05 to find for our t-value.

Our t-value needs to be above (2.447) to be able to *reject the null hypothesis*.

In [24]:
```
#This function finds the percentage differences between 2 column
#We will use this to compare trial store sales to control store scaled sales.
def pctdiff(dataset,trial,control,t_col,c_col):
    t = dataset.loc[dataset["STORE_NBR"] == trial,t_col].values
    c = dataset.loc[dataset["STORE_NBR"] == control,c_col].values
    return abs( t-c ) / ( (t + c) / 2 )
```

In [25]:
```
#This function turns scitific notation to decimal
def decimal_str(x: float, decimals: int = 10) -> str:
    return format(x, f".{decimals}f").lstrip().rstrip('0')
```

In [26]:
```
s77 = ( pctdiff(during_trial,77,control_77,"TOT_SALES","SCALED_SALES") - pctdiff(before_trial,77,contro
l_77,"TOT_SALES","SCALED_SALES").mean() )  / pctdiff(before_trial,77,control_77,"TOT_SALES","SCALED_SAL
ES").std()
print("Store 77 & 233 during trial period ( Feb2019-Apr2019 )")
print("t-value for each month : ",s77)
print("p-value for each month : ",decimal_str(stats.t.sf(s77[0],6)),decimal_str(stats.t.sf(s77[1],6)),d
ecimal_str(stats.t.sf(s77[2],6)))
```

```
Store 77 & 233 during trial period ( Feb2019-Apr2019 )
t-value for each month :  [-0.42290496  5.07070216  8.72031805]
p-value for each month :  0.656448729 0.0011432007 0.0000628666
```

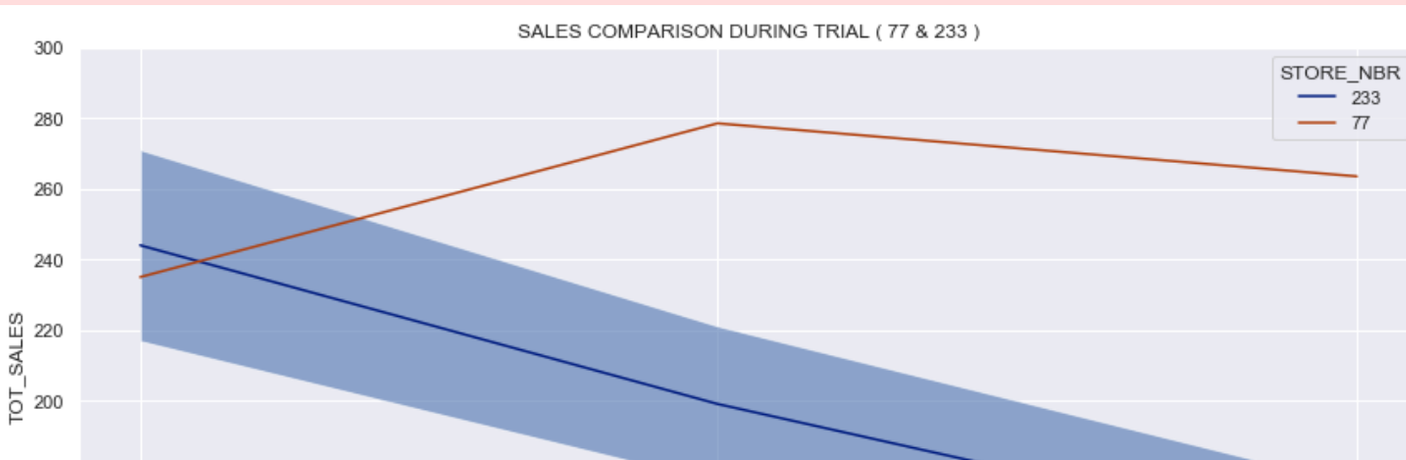
We can see that our p-value for month March & April is significantly less than our Alpha of 0.05 .
The plot below shows the sale of the control store 233 with its confidence interval and sales of the trial store 77 during the trial period.

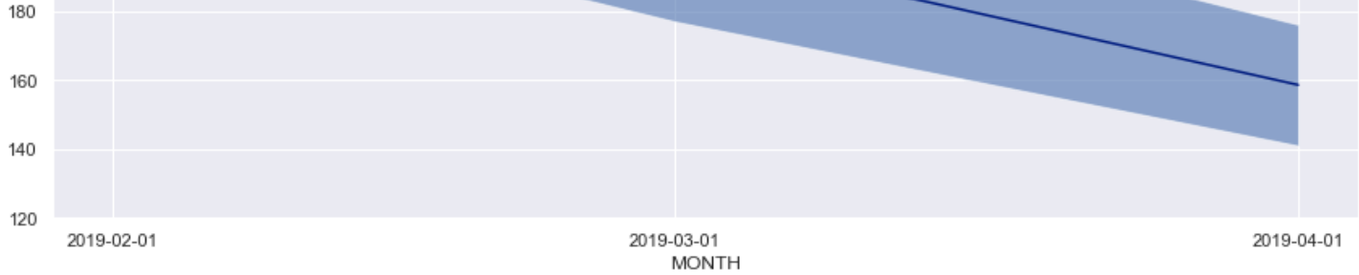
In [27]:
```
#Standard Deviation of the percentage of difference multiply by 2.447 (t value for 6 degree of freedom
 @ 95%)
#Use that value for out confidence interval. Value +- (CI_C77 * current value)
#CI_77 = std * t-value
CI_77 = pctdiff(before_trial,77,control_77,"TOT_SALES","SCALED_SALES").std() * 2.447
CI_U_77 = during_trial.loc[during_trial["STORE_NBR"] == control_77,"TOT_SALES"] + (during_trial.loc[dur
ing_trial["STORE_NBR"] == control_77,"TOT_SALES"] * CI_77 )
CI_D_77 = during_trial.loc[during_trial["STORE_NBR"] == control_77,"TOT_SALES"] - (during_trial.loc[dur
ing_trial["STORE_NBR"] == control_77,"TOT_SALES"] * CI_77 )
```

In [28]:
```
fig,axes = plt.subplots(figsize=(12,6))
fig.tight_layout()
sns.lineplot(x="MONTH",y="TOT_SALES",hue="STORE_NBR",hue_order=[control_77,77],data=during_trial.loc[du
ring_trial["STORE_NBR"].isin([77,control_77])],ax=axes,palette="dark")
plt.fill_between(during_trial["MONTH"].unique(),y1=CI_U_77,y2=CI_D_77,alpha=0.6)
axes.set_xticks(during_trial["MONTH"].unique())
axes.set_ylim(120,300)
axes.set_title("SALES COMPARISON DURING TRIAL ( 77 & 233 )")
plt.show()
```

```
c:\users\user\appdata\local\programs\python\python37-32\lib\site-packages\pandas\plotting\_matplotlib
\converter.py:103: FutureWarning: Using an implicitly registered datetime converter for a matplotlib
plotting method. The converter was registered by pandas on import. Future versions of pandas will req
uire you to explicitly register matplotlib converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```





Seeing that there is a statistical significant difference in sales during trial period we will check if this is similar for customer.

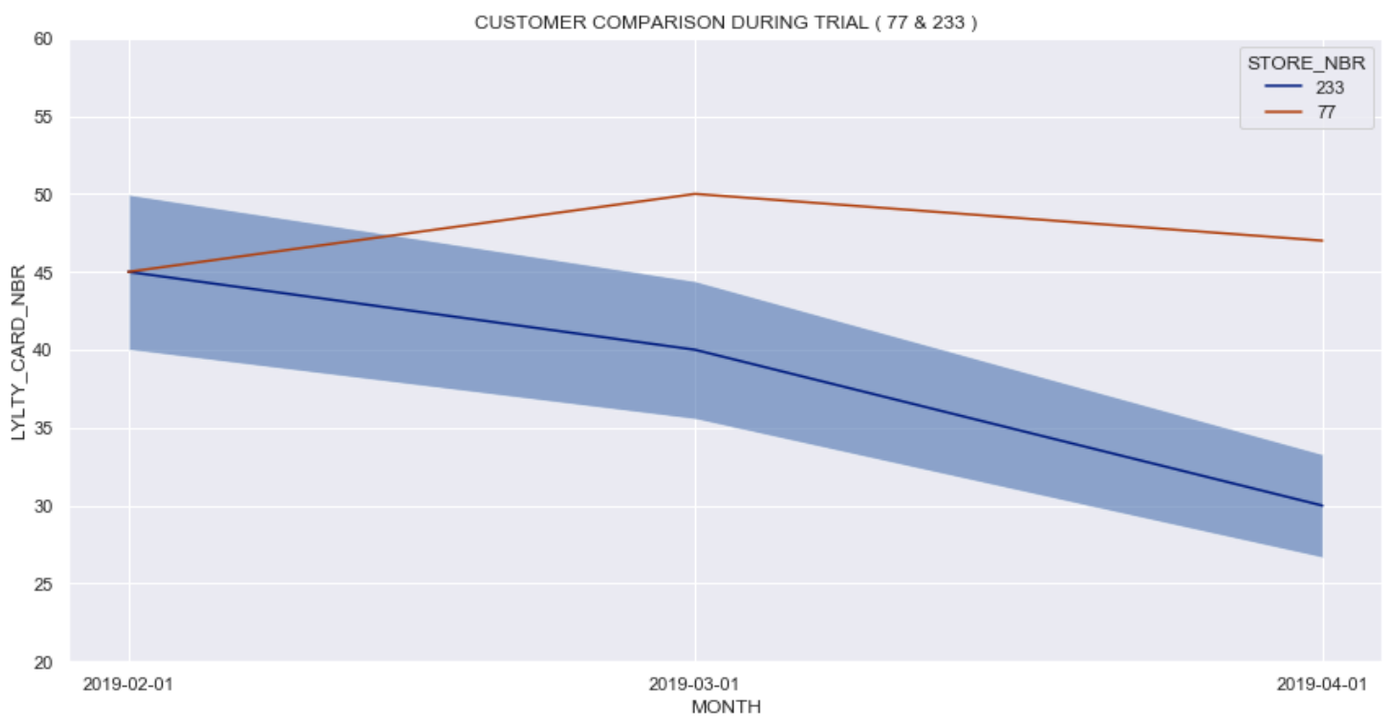
In [29]:
```
#T SCORE FOR LYLTY_CARD_NBR trial 77
c77 = ( pctdiff(during_trial,77,control_77,"LYLTY_CARD_NBR","SCALED_CUST") - pctdiff(before_trial,77,co
ntrol_77,"LYLTY_CARD_NBR","SCALED_CUST").mean() )  / pctdiff(before_trial,77,control_77,"LYLTY_CARD_NB
R","SCALED_CUST").std()
print("Store 77 & 233 during trial period ( Feb2019-Apr2019 )")
print("t-value for each month : ",c77)
print("p-value for each month : ",decimal_str(stats.t.sf(c77[0],6)),decimal_str(stats.t.sf(c77[1],6)),d
ecimal_str(stats.t.sf(c77[2],6)))
```

```
Store 77 & 233 during trial period ( Feb2019-Apr2019 )
t-value for each month :  [-0.947064   12.11487773 25.41282295]
p-value for each month :  0.8099148125 0.0000096086 0.0000001223
```

Here we can see that the p-value for 2 of the 3 month is significantly less than our Alpha level of 0.05 for customers. We can see this in the chart below that plots the confidence interval for the control store and customer for trial store.
Since we are measuring the percentage of difference. i will be using the percentage of difference for our confidence interval level. i.e: i take the standard deviation of the percentange of difference and multiply by 2.447(t value for 6 degree of freedom @ 95%). current value +- (current value * new percentage)

In [30]:
```
#Standard Deviation of the percentage of difference multiply by 2.447 (t value for 6 degree of freedom
 @ 95%)
#Use that value for out confidence interval. value +- (value * CI_C77 )
CI_C77 = pctdiff(before_trial,77,control_77,"LYLTY_CARD_NBR","SCALED_CUST").std() * 2.447
CI_U_C77 = during_trial.loc[during_trial["STORE_NBR"] == control_77,"LYLTY_CARD_NBR"] + (during_trial.l
oc[during_trial["STORE_NBR"] == control_77,"LYLTY_CARD_NBR"] * CI_77 )
CI_D_C77 = during_trial.loc[during_trial["STORE_NBR"] == control_77,"LYLTY_CARD_NBR"] - (during_trial.l
oc[during_trial["STORE_NBR"] == control_77,"LYLTY_CARD_NBR"] * CI_77 )
```

In [31]:
```
fig,axes = plt.subplots(figsize=(12,6))
fig.tight_layout()
sns.lineplot(x="MONTH",y="LYLTY_CARD_NBR",hue="STORE_NBR",hue_order=[control_77,77],data=during_trial.l
oc[during_trial["STORE_NBR"].isin([77,control_77])],ax=axes,palette="dark")
plt.fill_between(during_trial["MONTH"].unique(),y1=CI_U_C77,y2=CI_D_C77,alpha=0.6)
axes.set_xticks(during_trial["MONTH"].unique())
axes.set_ylim(20,60)
axes.set_title("CUSTOMER COMPARISON DURING TRIAL ( 77 & 233 )")
plt.show()
```



Now we will take a look at Trial Store 86 & Control Store 155 to see if there is any statistical significant against our null hypothesis.

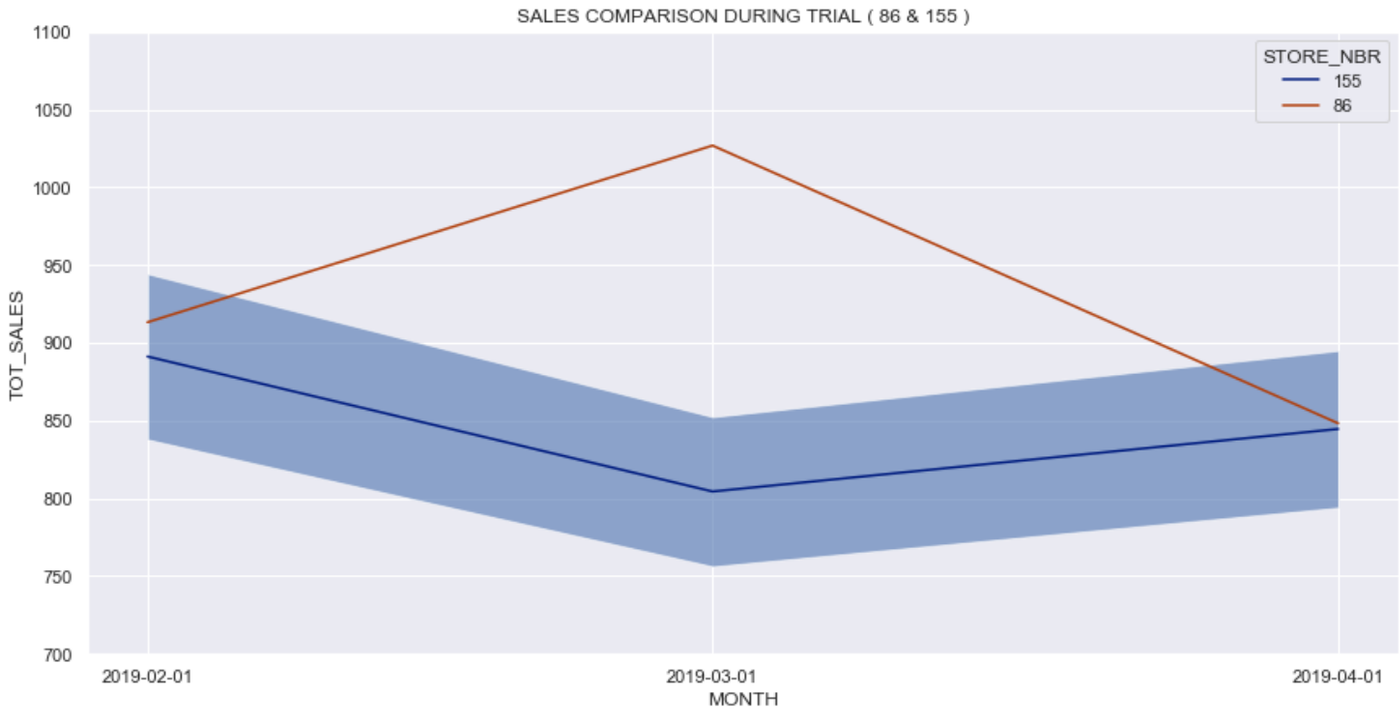
In [32]:
```
#t score for total sales trial 86
s86 = ( pctdiff(during_trial,86,control_86,"TOT_SALES","SCALED_SALES") - pctdiff(before_trial,86,contro
l_86,"TOT_SALES","SCALED_SALES").mean() )  / pctdiff(before_trial,86,control_86,"TOT_SALES","SCALED_SAL
ES").std()
print("Store 77 & 233 during trial period ( Feb2019-Apr2019 )")
print("t-value for each month : ",s86)
print("p-value for each month : ",decimal_str(stats.t.sf(s86[0],6)),decimal_str(stats.t.sf(s86[1],6)),d
ecimal_str(stats.t.sf(s86[2],6)))
```

```
Store 77 & 233 during trial period ( Feb2019-Apr2019 )
t-value for each month :  [ 1.19532112 10.12345264  0.37129535]
p-value for each month :  0.1385292664 0.0000270018 0.3615877376
```

It seems that only 1 of the 3 month during trial period has any statistical significant against the null hypothesis.
From the plot below we can see that for month Feb2019 & Apr2019 the sales for our trial store is within the confidence interval of our control store. And only March is significantly different between the 2 stores.

In [33]:
```
CI_86 = pctdiff(before_trial,86,control_86,"TOT_SALES","SCALED_SALES").std() * 2.447
CI_U_86 = during_trial.loc[during_trial["STORE_NBR"] == control_86,"TOT_SALES"] + (during_trial.loc[dur
ing_trial["STORE_NBR"] == control_86,"TOT_SALES"] * CI_86 )
CI_D_86 = during_trial.loc[during_trial["STORE_NBR"] == control_86,"TOT_SALES"] - (during_trial.loc[dur
ing_trial["STORE_NBR"] == control_86,"TOT_SALES"] * CI_86 )
```

In [34]:
```
fig,axes = plt.subplots(figsize=(12,6))
fig.tight_layout()
sns.lineplot(x="MONTH",y="TOT_SALES",hue="STORE_NBR",hue_order=[control_86,86],data=during_trial.loc[du
ring_trial["STORE_NBR"].isin([86,control_86])],ax=axes,palette="dark")
plt.fill_between(during_trial["MONTH"].unique(),y1=CI_U_86,y2=CI_D_86,alpha=0.6)
axes.set_xticks(during_trial["MONTH"].unique())
axes.set_ylim(700,1100)
axes.set_title("SALES COMPARISON DURING TRIAL ( 86 & 155 )")
plt.show()
```



Lets check if its similar for customers as well.

In [35]:
```
c86 = ( pctdiff(during_trial,86,control_86,"LYLTY_CARD_NBR","SCALED_CUST") - pctdiff(before_trial,86,co
ntrol_86,"LYLTY_CARD_NBR","SCALED_CUST").mean() ) / pctdiff(before_trial,86,control_86,"LYLTY_CARD_NBR"
,"SCALED_CUST").std()
print("Store 86 & 155 during trial period ( Feb2019-Apr2019 )")
print("t-value for each month : ",c86)
print("p-value for each month : ",decimal_str(stats.t.sf(c86[0],6)),decimal_str(stats.t.sf(c86[1],6)),d
ecimal_str(stats.t.sf(c86[2],6)))
```

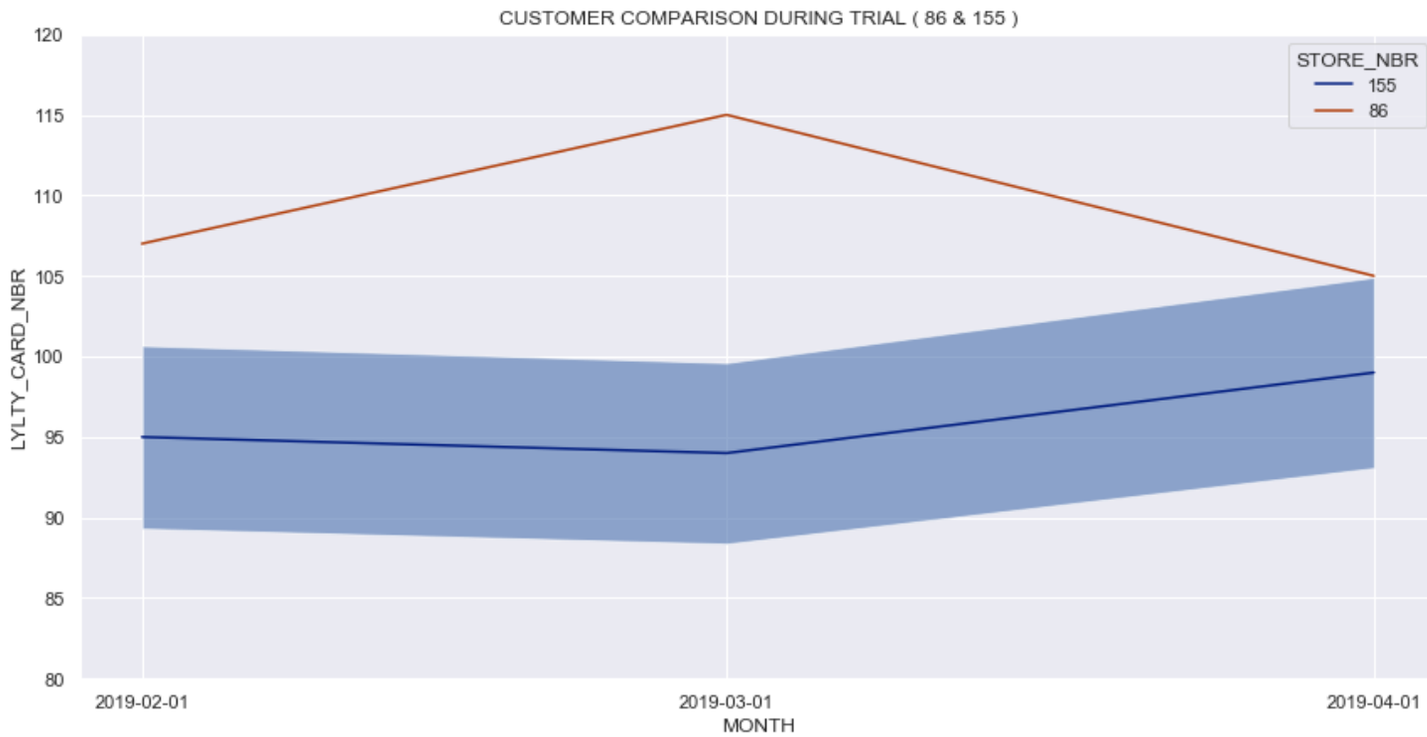
```
Store 86 & 155 during trial period ( Feb2019-Apr2019 )
t-value for each month :  [10.63090779 19.0188696   4.5054022 ]
p-value for each month :  0.0000204099 0.000000683 0.0020402007
```

Surpisingly the customer data shows that there is a significant increase in customer during the trial period. The p-value for all 3

month is significantly lower than our Alpha of .05 .
This could be due to a discount happening during the trial period that cause the sales data to be less impressive compared to customer data.
The increase in customer could be due to the discount or the new layout implemented during the trial period.
And from the graph below we can see that none of the 3 months is within our control stores confidence interval.

In [36]:
```
CI_C86 = pctdiff(before_trial,86,control_86,"LYLTY_CARD_NBR","SCALED_CUST").std() * 2.447
CI_U_C86 = during_trial.loc[during_trial["STORE_NBR"] == control_86,"LYLTY_CARD_NBR"] + ( during_trial.
loc[during_trial["STORE_NBR"] == control_86,"LYLTY_CARD_NBR"] * CI_86)
CI_D_C86 = during_trial.loc[during_trial["STORE_NBR"] == control_86,"LYLTY_CARD_NBR"] - ( during_trial.
loc[during_trial["STORE_NBR"] == control_86,"LYLTY_CARD_NBR"] * CI_86)
```

In [37]:
```
fig , axes = plt.subplots(figsize=(12,6))
fig.tight_layout()
sns.lineplot(x="MONTH",y="LYLTY_CARD_NBR",hue="STORE_NBR",hue_order=[control_86,86],data=during_trial.l
oc[during_trial["STORE_NBR"].isin([86,control_86])],ax=axes,palette="dark")
axes.set_xticks(during_trial["MONTH"].unique())
axes.set_ylim(80,120)
axes.set_title("CUSTOMER COMPARISON DURING TRIAL ( 86 & 155 )")
plt.fill_between(during_trial["MONTH"].unique(),y1=CI_U_C86,y2=CI_D_C86,alpha=0.6)
plt.show()
```



Lets move on to trial store 88.

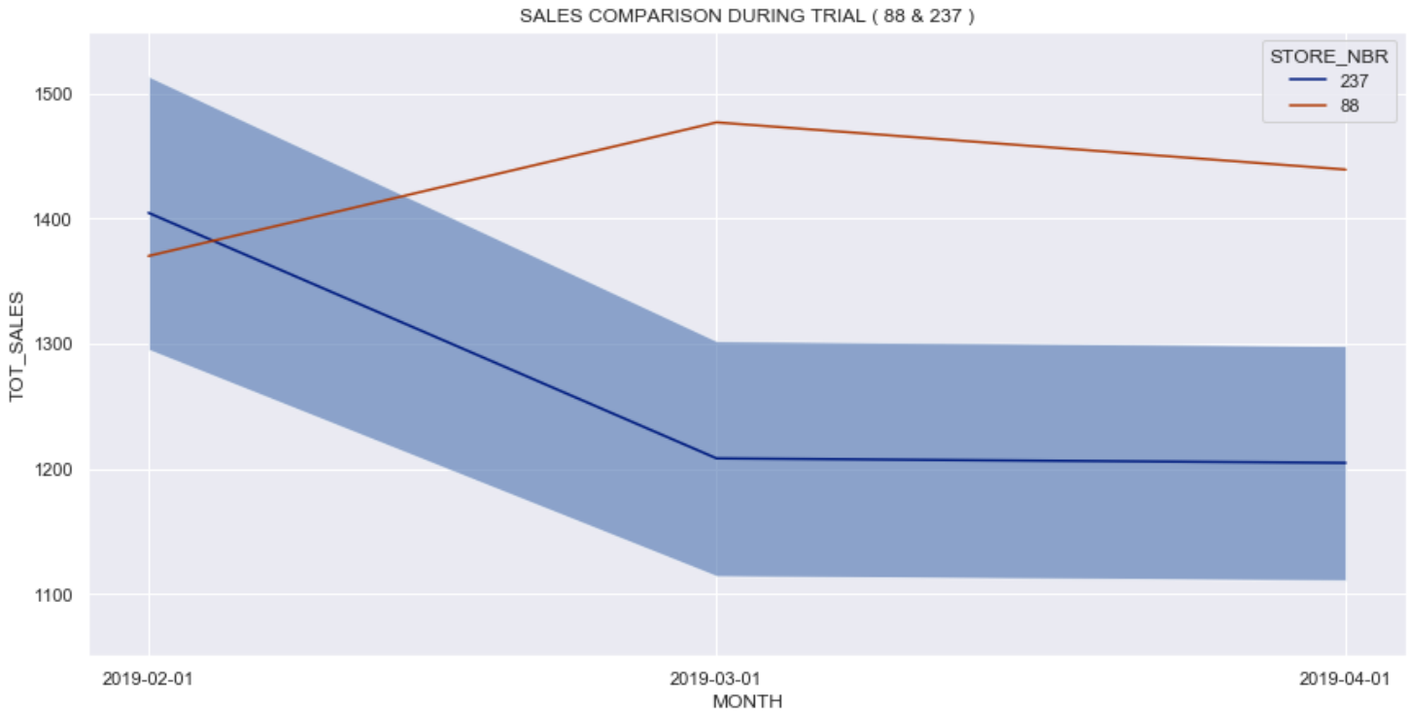
In [41]:
```
s88 =( pctdiff(during_trial,88,control_88,"TOT_SALES","SCALED_SALES") - pctdiff(before_trial,88,control
_88,"TOT_SALES","SCALED_SALES").mean() )/ pctdiff(before_trial,88,control_88,"TOT_SALES","SCALED_SALES"
).std()
print("Store 88 & 237 during trial period ( Feb2019-Apr2019 )")
print("t-value for each month : ",s88)
print("p-value for each month : ",decimal_str(stats.t.sf(s88[0],6)),decimal_str(stats.t.sf(s88[1],6)),d
ecimal_str(stats.t.sf(s88[2],6)))
```

```
Store 88 & 237 during trial period ( Feb2019-Apr2019 )
t-value for each month :  [-0.52553615  4.88301295  4.16934518]
p-value for each month :  0.6909710555 0.001378905 0.0029414129
```

Our sales data for trial store 88 shows 2 of the 3 months during trial period is statistically significant but by a thin margin. The p-value for March & April sits just below our Alpha of 0.05 .

In [43]:
```
CI_S88 = pctdiff(before_trial,88,control_88,"TOT_SALES","SCALED_SALES").std() * 2.447
CI_U_S88 = during_trial.loc[during_trial["STORE_NBR"] == control_88,"TOT_SALES"] + ( during_trial.loc[d
uring_trial["STORE_NBR"] == control_88,"TOT_SALES"] * CI_S88 )
CI_D_S88 = during_trial.loc[during_trial["STORE_NBR"] == control_88,"TOT_SALES"] - ( during_trial.loc[d
uring_trial["STORE_NBR"] == control_88,"TOT_SALES"] * CI_S88 )
```

In [54]:
```
fig, axes = plt.subplots(figsize=(12,6))
fig.tight_layout()
sns.lineplot(x="MONTH",y="TOT_SALES",hue="STORE_NBR",hue_order=[control_88,88],data=during_trial.loc[du
ring_trial["STORE_NBR"].isin([88,control_88])],ax=axes,palette="dark")
axes.set_title("SALES COMPARISON DURING TRIAL ( 88 & 237 )")
axes.set_xticks(during_trial["MONTH"].unique())
plt.fill_between(x=during_trial["MONTH"].unique(),y1=CI_U_S88,y2=CI_D_S88,alpha=0.6)
axes.set_ylim(1050,1550)
plt.show()
```



From the graph above we can see that statistically trial store 88 is similar to trial store 77 in terms of sales.
On to customer data.

In [55]:
```
c88 =( pctdiff(during_trial,88,control_88,"LYLTY_CARD_NBR","SCALED_CUST") - pctdiff(before_trial,88,con
trol_88,"LYLTY_CARD_NBR","SCALED_CUST").mean() )/ pctdiff(before_trial,88,control_88,"LYLTY_CARD_NBR",
"SCALED_CUST").std()
print("Store 88 & 237 during trial period ( Feb2019-Apr2019 )")
print("t-value for each month : ",c88)
print("p-value for each month : ",decimal_str(stats.t.sf(c88[0],6)),decimal_str(stats.t.sf(c88[1],6)),d
ecimal_str(stats.t.sf(c88[2],6)))
```

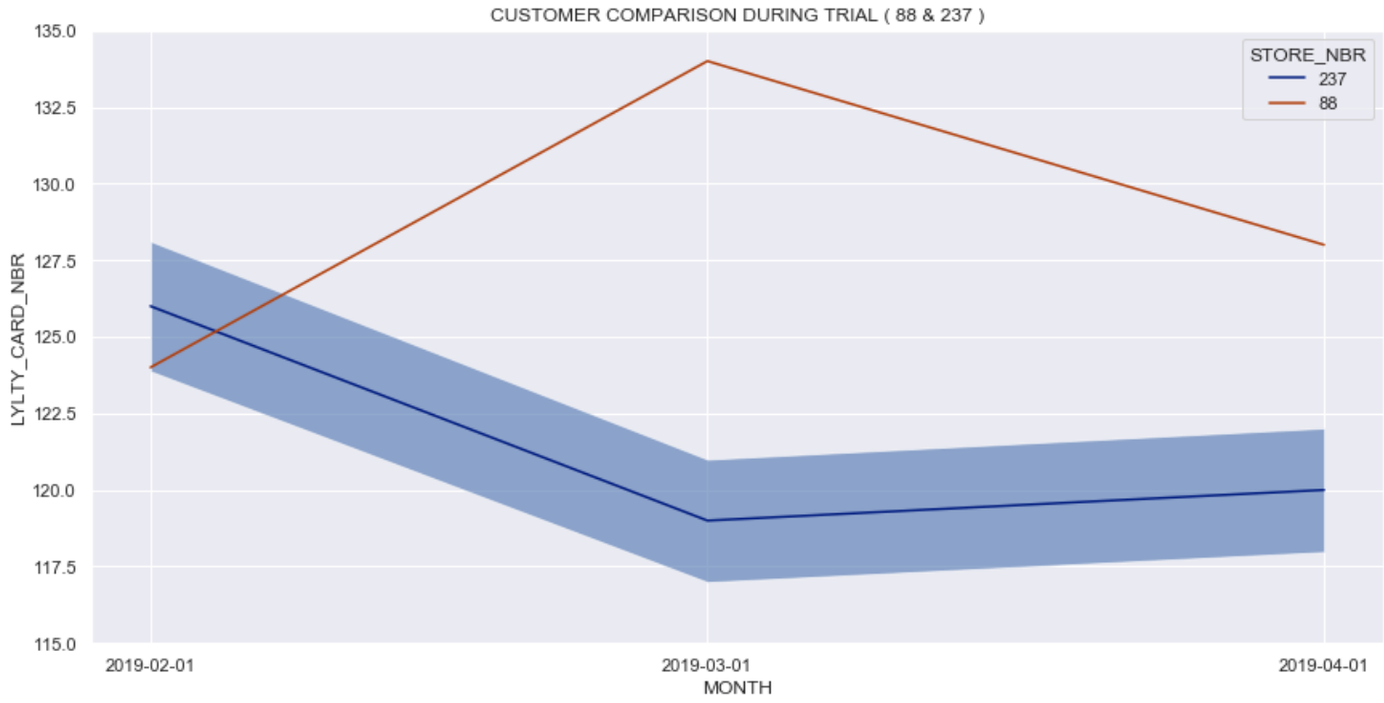
```
Store 88 & 237 during trial period ( Feb2019-Apr2019 )
t-value for each month :  [-0.38286932 16.1611498   8.31000844]
p-value for each month :  0.6424933304 0.0000017846 0.0000823347
```

The customer data for trial store 88 seems to show good correlation with sales data since 2 of the 3 months are statistically significant similar to our sales data.
We can check with the graph below to confirm our findings.

In [56]:
```
CI_C88 = pctdiff(before_trial,88,control_88,"LYLTY_CARD_NBR","SCALED_CUST").std() * 2.447
CI_U_C88 = during_trial.loc[during_trial["STORE_NBR"] == control_88,"LYLTY_CARD_NBR"]  + ( during_trial
.loc[during_trial["STORE_NBR"] == control_88,"LYLTY_CARD_NBR"] * CI_C88 )
CI_D_C88 = during_trial.loc[during_trial["STORE_NBR"] == control_88,"LYLTY_CARD_NBR"]  - ( during_trial
.loc[during_trial["STORE_NBR"] == control_88,"LYLTY_CARD_NBR"] * CI_C88 )
```

In [63]:
```
fig,axes = plt.subplots(figsize=(12,6))
fig.tight_layout()

sns.lineplot(x="MONTH",y="LYLTY_CARD_NBR",hue="STORE_NBR",hue_order=[control_88,88],data=during_trial.l
oc[during_trial["STORE_NBR"].isin([88,control_88])],ax=axes,palette="dark")
axes.set_title("CUSTOMER COMPARISON DURING TRIAL ( 88 & 237 )")
axes.set_ylim(115,135)
axes.set_xticks(during_trial["MONTH"].unique())
plt.fill_between(during_trial["MONTH"].unique(),y1=CI_U_C88,y2=CI_D_C88,alpha=0.6)
plt.show()
```



CONCLUSION:

The results for trial store 77 and 88 during the trial period shows a significant difference in 2 of the 3 trial months. As for store 86 the data shows significant increase in customers but the sales data seems to be lacking. This could be caused by some other factor like a discount or sales was held during the trial period.*Overall the new layout shows a significant increase in sales and customer.*

In [ ]: