Quantium task 2

August 9, 2021

1 Quantium Virtual internship

Task 2

1.1 Experimentation and uplift testing

Extend your analysis from Task 1 to help you identify benchmark stores that allow you to test the impact of the trial store layouts on customer sales.

To get started use the QVI_data dataset below or your output from task 1 and consider the monthly sales experience of each store.

This can be broken down by:

- Total sales revenue
- Total number of customers
- Average number of transactions per customer

Create a measure to compare different control stores to each of the trial stores to do this write a function to reduce having to re-do the analysis for each trial store. Consider using Pearson correlations or a metric such as a magnitude distance e.g. 1- (Observed distance – minimum distance)/(Maximum distance – minimum distance) as a measure.

Once you have selected your control stores, compare each trial and control pair during the trial period. You want to test if total sales are significantly different in the trial period and if so, check if the driver of change is more purchasing customers or more purchases per customers etc.

For this part of the project we will be examining the performance in trial vs control stores to provide a recommendation for each location based on our insight.

Key of solution:-

- Consider of monthly sales so we have to check dates pd.to_datetime()
- To consider monthly sales we have to broke down data by:-
  - Total Sales revenue per month
  - Total number of customers per month
  - Average number of transactions per customer per month
- Create a measure to compare different control stores to each of the trial stores
  - Write a function to reduce having to re-do the analysis for each trial store.

- Consider using Pearson correlations or a metric such as a magnitude distance link to pearson correlation
- Compare each trial and control pair during the trial period.
- Test if total sales are significantly different in the trial period
- Check if the driver of change is more purchasing customers or more purchases per customers

- evaluate the performance of a store trial which was performed in stores 77, 86 and 88.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
# warnings.filterwarnings(action='once')
```

[2]:
```
dataset = pd.read_csv("QVI_data.csv")
dataset.head()
```

[2]:
```
   LYLTY_CARD_NBR        DATE  STORE_NBR  TXN_ID  PROD_NBR  \
0            1000  2018-10-17          1       1         5
1            1002  2018-09-16          1       2        58
2            1003  2019-03-07          1       3        52
3            1003  2019-03-08          1       4       106
4            1004  2018-11-02          1       5        96

                                PROD_NAME  PROD_QTY  TOT_SALES  PACK_SIZE  \
0  Natural Chip        Compny SeaSalt175g         2        6.0        175
1   Red Rock Deli Chikn&Garlic Aioli 150g         1        2.7        150
2   Grain Waves Sour    Cream&Chives 210G         1        3.6        210
3  Natural ChipCo      Hony Soy Chckn175g         1        3.0        175
4          WW Original Stacked Chips 160g         1        1.9        160

        BRAND              LIFESTAGE PREMIUM_CUSTOMER
0     NATURAL  YOUNG SINGLES/COUPLES          Premium
1         RRD  YOUNG SINGLES/COUPLES       Mainstream
2     GRNWVES         YOUNG FAMILIES           Budget
3     NATURAL         YOUNG FAMILIES           Budget
4  WOOLWORTHS  OLDER SINGLES/COUPLES       Mainstream
```

[3]:
```
## check columns types
dataset.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
```

[4]:
```
## We will convert Date column to_datetime column
dataset['DATE']=pd.to_datetime(dataset['DATE'])
```

[5]:
```
## Next we will create a new column contain number of year + number of month
dataset['YEAR-MONTH']=[s.year*100+s.month for s in dataset['DATE']]
```

[6]:
```
dataset.head()
```

[6]:
```
   LYLTY_CARD_NBR       DATE  STORE_NBR  TXN_ID  PROD_NBR  \
0            1000 2018-10-17          1       1         5
1            1002 2018-09-16          1       2        58
2            1003 2019-03-07          1       3        52
3            1003 2019-03-08          1       4       106
4            1004 2018-11-02          1       5        96

                                  PROD_NAME  PROD_QTY  TOT_SALES  PACK_SIZE  \
0  Natural Chip        Compny SeaSalt175g         2        6.0        175
1   Red Rock Deli Chikn&Garlic Aioli 150g         1        2.7        150
2   Grain Waves Sour    Cream&Chives 210G         1        3.6        210
3  Natural ChipCo      Hony Soy Chckn175g         1        3.0        175
4          WW Original Stacked Chips 160g         1        1.9        160

        BRAND              LIFESTAGE PREMIUM_CUSTOMER  YEAR-MONTH
0     NATURAL  YOUNG SINGLES/COUPLES          Premium      201810
1         RRD  YOUNG SINGLES/COUPLES       Mainstream      201809
2     GRNWVES         YOUNG FAMILIES           Budget      201903
3     NATURAL         YOUNG FAMILIES           Budget      201903
4  WOOLWORTHS  OLDER SINGLES/COUPLES       Mainstream      201811
```

[7]:
```
dataset.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  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
 12  YEAR-MONTH        264834 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(7), object(4)
memory usage: 26.3+ MB
```

1.1.1 Grouping and Aggregating - Analyzing and Exploring Our Data

You can learn more from here

We will also need to count the unique values of each column of a dataframe, we will use the pandas dataframe **nunique()** function.

- The following is the syntax:
  - counts = df.nunique()

Here, df is the dataframe for which you want to know the unique counts. It returns a pandas Series of counts. By default, the pandas dataframe nunique() function counts the distinct values along axis=0, that is, row-wise which gives you the count of distinct values in each column. Source

[8]:
```python
counts = dataset.nunique()
print(counts)
```

```
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
```

```
YEAR-MONTH                12
dtype: int64
```

So from our counts we saw that there's:- - 72636 card numbers - 364 Date - 272 Stores - 263125 ID - 114 Products - 5 main counts for quantity - 111 number of total sales - 21 Brands - 21 Package sizes - 7 types of lifestages - 3 types of premium customers - 12 numbers of our newly created column year-month

[9]:
```python
print(dataset['PROD_QTY'].value_counts())
print(dataset['PROD_QTY'].value_counts(normalize = True))
```

```
2    236039
1     27518
5       450
3       430
4       397
Name: PROD_QTY, dtype: int64
2    0.891272
1    0.103907
5    0.001699
3    0.001624
4    0.001499
Name: PROD_QTY, dtype: float64
```

[10]:
```python
print(dataset['BRAND'].value_counts())
print(dataset['BRAND'].value_counts(normalize= True))
```

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

```
KETTLE        0.155901
SMITHS        0.120162
DORITOS       0.106274
PRINGLES      0.094784
RRD           0.067133
WOOLWORTHS    0.055722
INFUZIONS     0.053622
THINS         0.053146
COBS          0.036600
TOSTITOS      0.035762
TWISTIES      0.035698
OLD           0.035207
GRNWVES       0.029226
NATURAL       0.028203
TYRRELLS      0.024325
CHEEZELS      0.017381
CCS           0.017184
SUNBITES      0.011358
CHEETOS       0.011052
BURGER        0.005906
FRENCH        0.005354
Name: BRAND, dtype: float64
```

[11]:

```
print(dataset['PACK_SIZE'].value_counts())
print(dataset['PACK_SIZE'].value_counts(normalize = True))
```

```
175    66390
150    43131
134    25102
110    22387
170    19983
165    15297
300    15166
330    12540
380     6416
270     6285
210     6272
200     4473
135     3257
250     3169
90      3008
190     2995
160     2970
220     1564
70      1507
180     1468
125     1454
Name: PACK_SIZE, dtype: int64
```

```
175    0.250685
150    0.162861
134    0.094784
110    0.084532
170    0.075455
165    0.057761
300    0.057266
330    0.047350
380    0.024226
270    0.023732
210    0.023683
200    0.016890
135    0.012298
250    0.011966
90     0.011358
190    0.011309
160    0.011215
220    0.005906
70     0.005690
180    0.005543
125    0.005490
Name: PACK_SIZE, dtype: float64
```

[12]:
```
print(dataset['LIFESTAGE'].value_counts())
print(dataset['LIFESTAGE'].value_counts(normalize= True))
```

```
OLDER SINGLES/COUPLES     54479
RETIREES                  49763
OLDER FAMILIES            48594
YOUNG FAMILIES            43592
YOUNG SINGLES/COUPLES     36377
MIDAGE SINGLES/COUPLES    25110
NEW FAMILIES               6919
Name: LIFESTAGE, dtype: int64
OLDER SINGLES/COUPLES     0.205710
RETIREES                  0.187903
OLDER FAMILIES            0.183489
YOUNG FAMILIES            0.164601
YOUNG SINGLES/COUPLES     0.137358
MIDAGE SINGLES/COUPLES    0.094814
NEW FAMILIES              0.026126
Name: LIFESTAGE, dtype: float64
```

[13]:
```
print(dataset['PREMIUM_CUSTOMER'].value_counts())
print(dataset['PREMIUM_CUSTOMER'].value_counts(normalize = True))
```

```
Mainstream    101988
Budget         93157
Premium        69689
```

```
Name: PREMIUM_CUSTOMER, dtype: int64
Mainstream    0.385102
Budget        0.351756
Premium       0.263142
Name: PREMIUM_CUSTOMER, dtype: float64
```

Premium customer types interpretation:- - Mainstream 101988 person with 38.5% of total data - Budget 93157 person with 35.2% of total data - Premium 69689 person with 26.3% of total data

[14]:
```
print(dataset['YEAR-MONTH'].value_counts()) ## to get percentage we will use␣
 ↪normalize = True
print(dataset['YEAR-MONTH'].value_counts(normalize = True))
```

```
201812    22835
201903    22592
201807    22562
201808    22410
201905    22391
201810    22288
201901    22161
201811    21852
201906    21829
201904    21766
201809    21743
201902    20405
Name: YEAR-MONTH, dtype: int64
201812    0.086224
201903    0.085306
201807    0.085193
201808    0.084619
201905    0.084547
201810    0.084158
201901    0.083679
201811    0.082512
201906    0.082425
201904    0.082187
201809    0.082100
201902    0.077048
Name: YEAR-MONTH, dtype: float64
```

YEAR-MONTH interpretation:- - 201812 —> 2018 - 12 there's 22835 transaction with 8.622% of total data - 201903 —> 2019 - 03 there's 22592 transaction with 8.530% of total data - 201807 —> 2018 - 07 there's 22562 transaction with 8.519% of total data - 201808 —> 2018 - 08 there's 22410 transaction with 8.461% of total data - 201905 —> 2019 - 05 there's 22391 transaction with 8.454% of total data - 201810 —> 2018 - 10 there's 22288 transaction with 8.415% of total data - 201901 —> 2019 - 01 there's 22161 transaction with 8.367% of total data - 201811 —> 2018 - 11 there's 21852 transaction with 8.251% of total data - 201906 —> 2019 - 06 there's 21829 transaction with 8.242% of total data - 201904 —> 2019 - 04 there's 21766 transaction with 8.218% of total data - 201809 —> 2018 - 09 there's 21743 transaction with 8.210% of total data - 201902 —> 2019 - 02 there's 20405 transaction with 7.704% of total data

[15]:
```
dataset.describe()
```

[15]:
```
       LYLTY_CARD_NBR      STORE_NBR        TXN_ID       PROD_NBR  \
count    2.648340e+05  264834.000000  2.648340e+05  264834.000000
mean     1.355488e+05     135.079423  1.351576e+05      56.583554
std      8.057990e+04      76.784063  7.813292e+04      32.826444
min      1.000000e+03       1.000000  1.000000e+00       1.000000
25%      7.002100e+04      70.000000  6.760050e+04      28.000000
50%      1.303570e+05     130.000000  1.351365e+05      56.000000
75%      2.030940e+05     203.000000  2.026998e+05      85.000000
max      2.373711e+06     272.000000  2.415841e+06     114.000000

            PROD_QTY      TOT_SALES      PACK_SIZE     YEAR-MONTH
count  264834.000000  264834.000000  264834.000000  264834.000000
mean        1.905813       7.299346     182.425512  201856.055163
std         0.343436       2.527241      64.325148      47.035278
min         1.000000       1.500000      70.000000  201807.000000
25%         2.000000       5.400000     150.000000  201809.000000
50%         2.000000       7.400000     170.000000  201812.000000
75%         2.000000       9.200000     175.000000  201903.000000
max         5.000000      29.500000     380.000000  201906.000000
```

[16]:
```
dataset.isnull().sum()
```

[16]:
```
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
YEAR-MONTH          0
dtype: int64
```

Groupby operation is **splitting** the object, **Applying** a function and **combining** the results

Aggregation if we want to run multiple aggregate functions on each column like sum, nunique, mean and so on.

[17]:
```
metrics = dataset.groupby(['STORE_NBR','YEAR-MONTH']).agg({'LYLTY_CARD_NBR':
 ↪'nunique', 'TXN_ID':'nunique', 'PROD_QTY':'sum', 'TOT_SALES':'sum'})
metrics['PRICE_PER_UNIT']=metrics['TOT_SALES']/metrics['PROD_QTY']
```

```
metrics['CHIP_PER_TXN']=metrics['PROD_QTY']/metrics['TXN_ID']
metrics=metrics.rename(columns={'LYLTY_CARD_NBR':'CUSTOMERS'})
metrics['TXN_PER_CUST']=metrics['TXN_ID']/metrics['CUSTOMERS']
metrics.drop(['TXN_ID'],axis=1,inplace=True)
```

[18]:
```
mod = metrics.copy()
```

[19]:
```
#taking data before 2019-02 into consideration
trial_stores=[]
for i in metrics.index:
    if(i[1]>=201902):
        if(i[1]<=201904):
            trial_stores.append(metrics.loc[i])
        metrics.drop(i,inplace=True)
trial_stores=pd.DataFrame(trial_stores)
```

[20]:
```
#we will do th same for data after 2019-02 into trial dataframe
trial_stores.index.name=('IDX')
k=0
trial_stores['STORE_NBR']=0
trial_stores['MONTHYEAR']=0
for (i,j) in trial_stores.reset_index()['IDX']:
    trial_stores['STORE_NBR'].iloc[k]=i
    trial_stores['MONTHYEAR'][k]=j
    k=k+1
trial_stores=trial_stores.set_index(['STORE_NBR','MONTHYEAR'])
```

[21]:
```
metrics.head(15)
```

[21]:
```
                     CUSTOMERS  PROD_QTY  TOT_SALES  PRICE_PER_UNIT  \
STORE_NBR YEAR-MONTH
1         201807            49        62      206.9        3.337097
          201808            42        54      176.1        3.261111
          201809            59        75      278.8        3.717333
          201810            44        58      188.1        3.243103
          201811            46        57      192.6        3.378947
          201812            42        57      189.6        3.326316
          201901            35        42      154.8        3.685714
2         201807            39        46      150.8        3.278261
          201808            39        55      193.8        3.523636
          201809            36        41      154.4        3.765854
          201810            41        45      167.8        3.728889
          201811            39        44      162.9        3.702273
          201812            35        40      136.0        3.400000
          201901            43        49      162.8        3.322449
3         201807           112       271     1205.7        4.449077
```

```
                      CHIP_PER_TXN  TXN_PER_CUST
STORE_NBR YEAR-MONTH
1         201807          1.192308      1.061224
          201808          1.255814      1.023810
          201809          1.209677      1.050847
          201810          1.288889      1.022727
          201811          1.212766      1.021739
          201812          1.212766      1.119048
          201901          1.166667      1.028571
2         201807          1.121951      1.051282
          201808          1.279070      1.102564
          201809          1.108108      1.027778
          201810          1.046512      1.048780
          201811          1.100000      1.025641
          201812          1.081081      1.057143
          201901          1.088889      1.046512
3         201807          1.963768      1.232143
```

[24]:

```python
# Now we will write some Functions to find correlation and magnitude of stores␣
 ↪with each other

def calcCorr(store):
    '''
    input=store number which is to be compared
    output=dataframe with corelation coefficient values
    '''
    a=[]
    #add metrics as required e.g. ,'TXN_PER_CUST'
    matrix=metrics[['TOT_SALES','CUSTOMERS']]
    for i in matrix.index:
        a.append(matrix.loc[store].corrwith(matrix.loc[i[0]]))
    df= pd.DataFrame(a)
    df.index=matrix.index
    df=df.drop_duplicates()
    df.index=[s[0] for s in df.index]
    df.index.name="STORE_NBR"
    return df
```

[29]:

```python
def standardizer(df):
    '''
    input=dataframe with metrics
    output=dataframe with mean of the metrics in a new column
    '''
    df=df.abs()
    df['MAGNITUDE']=df.mean(axis=1)
    return df
```

Our head of data science team asked us to evaluate the performance of a store trial which was performed in stores 77, 86 and 88.

[35]:
```
calc_corr_77 = calcCorr(77)
calc_corr_77
```

[35]:
```
           TOT_SALES  CUSTOMERS
STORE_NBR
1           0.075218   0.322168
2          -0.263079  -0.572051
3           0.806644   0.834207
4          -0.263300  -0.295639
5          -0.110652   0.370659
…                …          …
268         0.344757   0.369517
269        -0.315730  -0.474293
270         0.315430  -0.131259
271         0.355487   0.019629
272         0.117622   0.223217

[266 rows x 2 columns]
```

[36]:
```
calc_corr_77 = standardizer(calc_corr_77)
calc_corr_77
```

[36]:
```
           TOT_SALES  CUSTOMERS  MAGNITUDE
STORE_NBR
1           0.075218   0.322168   0.198693
2          -0.263079  -0.572051  -0.417565
3           0.806644   0.834207   0.820426
4          -0.263300  -0.295639  -0.279469
5          -0.110652   0.370659   0.130003
…                …          …          …
268         0.344757   0.369517   0.357137
269        -0.315730  -0.474293  -0.395011
270         0.315430  -0.131259   0.092086
271         0.355487   0.019629   0.187558
272         0.117622   0.223217   0.170420

[266 rows x 3 columns]
```

[38]:
```
calc_corr_77=calc_corr_77.sort_values(['MAGNITUDE'],ascending=False).dropna()
calc_corr_77
```

[38]:
```
           TOT_SALES  CUSTOMERS  MAGNITUDE
STORE_NBR
77          1.000000   1.000000   1.000000
233         0.903774   0.990358   0.947066
119         0.867664   0.983267   0.925466
```

```
71          0.914106   0.754817   0.834461
3           0.806644   0.834207   0.820426
…              …          …          …
19         -0.677929  -0.633453  -0.655691
242        -0.692664  -0.643351  -0.668008
75         -0.806751  -0.590735  -0.698743
186        -0.820214  -0.635966  -0.728090
9          -0.702976  -0.785699  -0.744338

[263 rows x 3 columns]
```

- As we can see from here store 233 has the heighest correlation with store 77

[39]:
```python
calc_corr_86 = calcCorr(86)
calc_corr_86
```

[39]:
```
           TOT_SALES  CUSTOMERS
STORE_NBR
1           0.445632   0.485831
2          -0.403835  -0.086161
3          -0.261284  -0.353786
4          -0.039035  -0.169608
5           0.235159  -0.253229
…              …          …
268        -0.452182  -0.034273
269         0.697055  -0.098587
270        -0.730679  -0.767267
271         0.527637   0.267393
272         0.004926  -0.353815

[266 rows x 2 columns]
```

[40]:
```python
calc_corr_86 = standardizer(calc_corr_86)
calc_corr_86
```

[40]:
```
           TOT_SALES  CUSTOMERS  MAGNITUDE
STORE_NBR
1           0.445632   0.485831   0.465731
2          -0.403835  -0.086161  -0.244998
3          -0.261284  -0.353786  -0.307535
4          -0.039035  -0.169608  -0.104322
5           0.235159  -0.253229  -0.009035
…              …          …          …
268        -0.452182  -0.034273  -0.243228
269         0.697055  -0.098587   0.299234
270        -0.730679  -0.767267  -0.748973
271         0.527637   0.267393   0.397515
272         0.004926  -0.353815  -0.174445
```

```
[266 rows x 3 columns]
```

[41]:
```
calc_corr_86=calc_corr_86.sort_values(['MAGNITUDE'],ascending=False).dropna()
calc_corr_86
```

[41]:
```
           TOT_SALES  CUSTOMERS  MAGNITUDE
STORE_NBR
86          1.000000   1.000000   1.000000
155         0.877882   0.942876   0.910379
114         0.734415   0.855339   0.794877
260         0.720350   0.846502   0.783426
109         0.788300   0.770778   0.779539
…              …          …         …
270        -0.730679  -0.767267  -0.748973
185        -0.776923  -0.741749  -0.759336
108        -0.840413  -0.697245  -0.768829
120        -0.872693  -0.815097  -0.843895
23         -0.784698  -0.943559  -0.864128

[263 rows x 3 columns]
```

- As we can see from here store 155 has the heighest correlation with store 86

[42]:
```
calc_corr_88 = calcCorr(88)
calc_corr_88
```

[42]:
```
           TOT_SALES  CUSTOMERS
STORE_NBR
1           0.813636   0.305334
2          -0.067927  -0.452379
3          -0.507847   0.522884
4          -0.745566  -0.361503
5           0.190330  -0.025320
…              …          …
268        -0.021429   0.672672
269        -0.172578  -0.274781
270        -0.723272  -0.103032
271        -0.103037  -0.018831
272        -0.772772   0.026909

[266 rows x 2 columns]
```

[43]:
```
calc_corr_88 = standardizer(calc_corr_88)
calc_corr_88
```

[43]:
```
           TOT_SALES  CUSTOMERS  MAGNITUDE
STORE_NBR
```

```
1            0.813636   0.305334   0.559485
2           -0.067927  -0.452379  -0.260153
3           -0.507847   0.522884   0.007518
4           -0.745566  -0.361503  -0.553534
5            0.190330  -0.025320   0.082505
…                 …          …          …
268         -0.021429   0.672672   0.325621
269         -0.172578  -0.274781  -0.223679
270         -0.723272  -0.103032  -0.413152
271         -0.103037  -0.018831  -0.060934
272         -0.772772   0.026909  -0.372932

[266 rows x 3 columns]
```

[44]:
```
calc_corr_88=calc_corr_88.sort_values(['MAGNITUDE'],ascending=False).dropna()
calc_corr_88
```

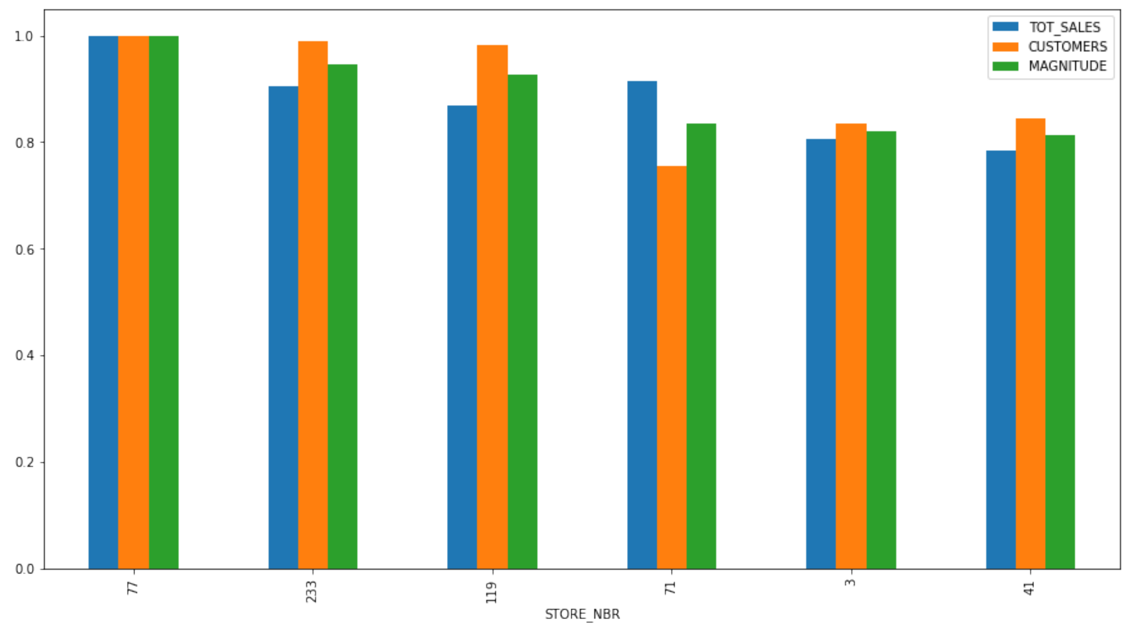
[44]:
```
           TOT_SALES  CUSTOMERS  MAGNITUDE
STORE_NBR
88          1.000000   1.000000   1.000000
178         0.731857   0.939466   0.835661
14          0.698557   0.942976   0.820767
204         0.885774   0.550263   0.718018
134         0.864293   0.508880   0.686587
…                 …          …          …
48         -0.857142  -0.361505  -0.609324
141        -0.690590  -0.547399  -0.618994
227        -0.537448  -0.729943  -0.633695
239        -0.642329  -0.660672  -0.651501
133        -0.735407  -0.835426  -0.785417

[263 rows x 3 columns]
```

- As we can see from here store 178 has the heighest correlation with store 88

We will start working and visualizing our data from 3 stores so the same steps we will do for tail store 77 and its control store 233 will be replicated.

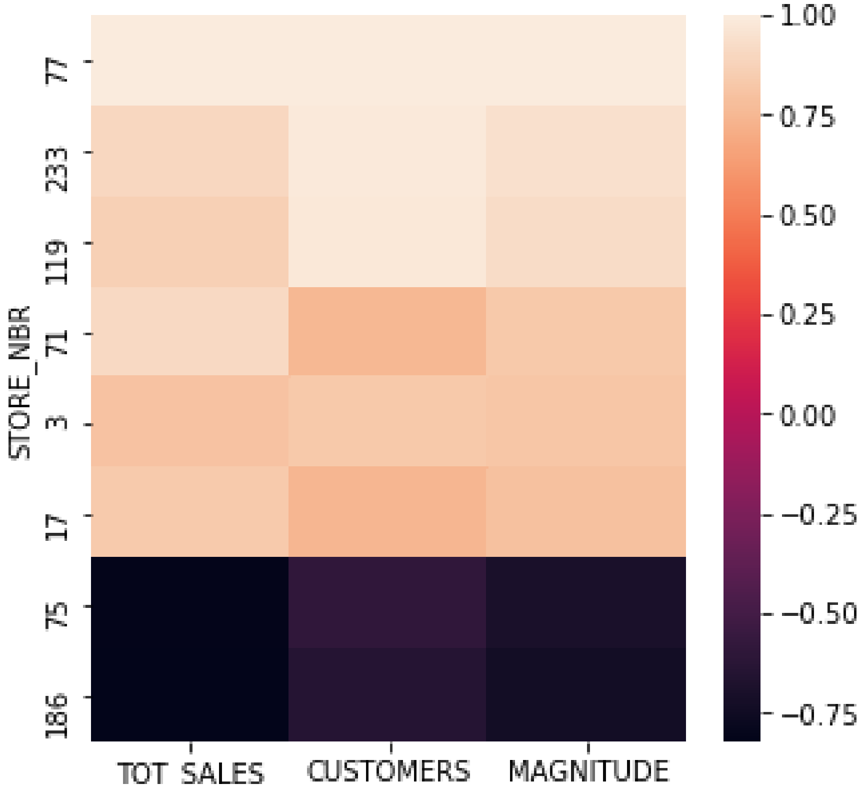
[48]:
```
#Taking 0.8 as threshold correlation
calc_corr_77[(calc_corr_77.MAGNITUDE.abs()>0.8)].plot(kind='bar',figsize=(15,8))
```

[48]:
```
<AxesSubplot:xlabel='STORE_NBR'>
```



[54]:
```
plt.figure(figsize=(5,5))
sns.heatmap(calc_corr_77[calc_corr_77.TOT_SALES.abs()>0.8])
```

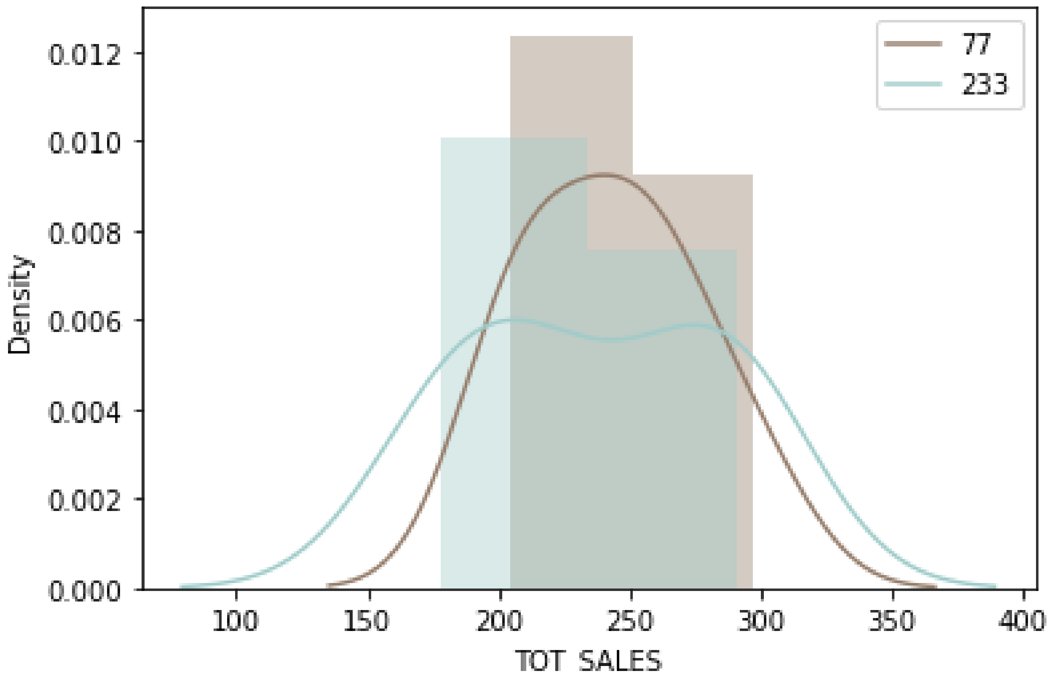
[54]: <AxesSubplot:ylabel='STORE_NBR'>



Taking store 233 into consideration plotting different measure against those of store 77

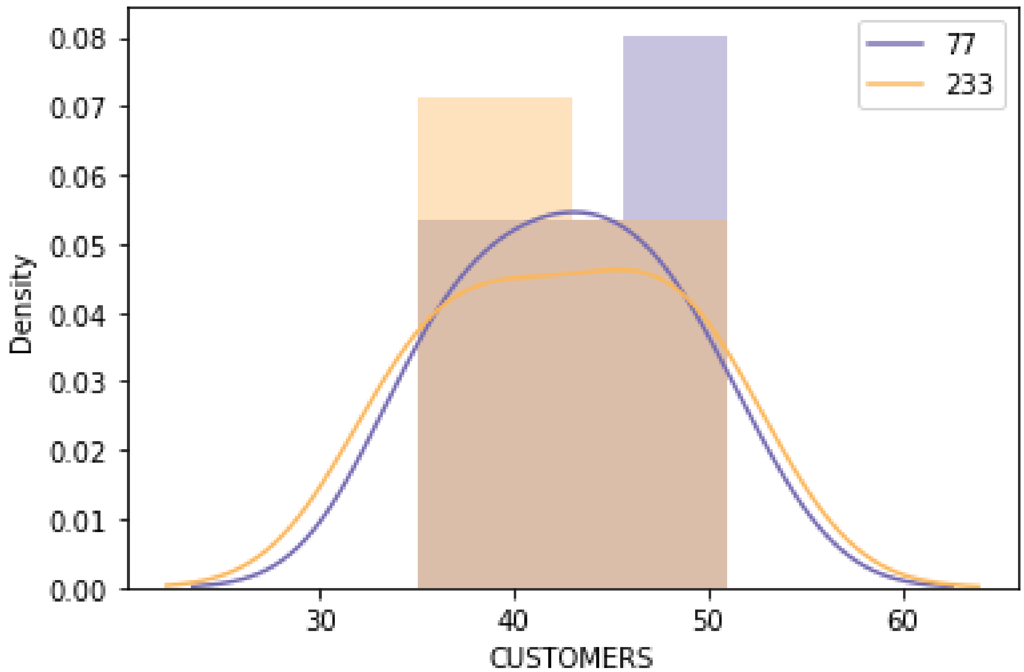
[66]:
```
sns.distplot(metrics.loc[77]['TOT_SALES'], color='#957d6b')
sns.distplot(metrics.loc[233]['TOT_SALES'], color='#a0ccca')
plt.legend(labels=['77','233'])
```

[66]: <matplotlib.legend.Legend at 0x5754280>



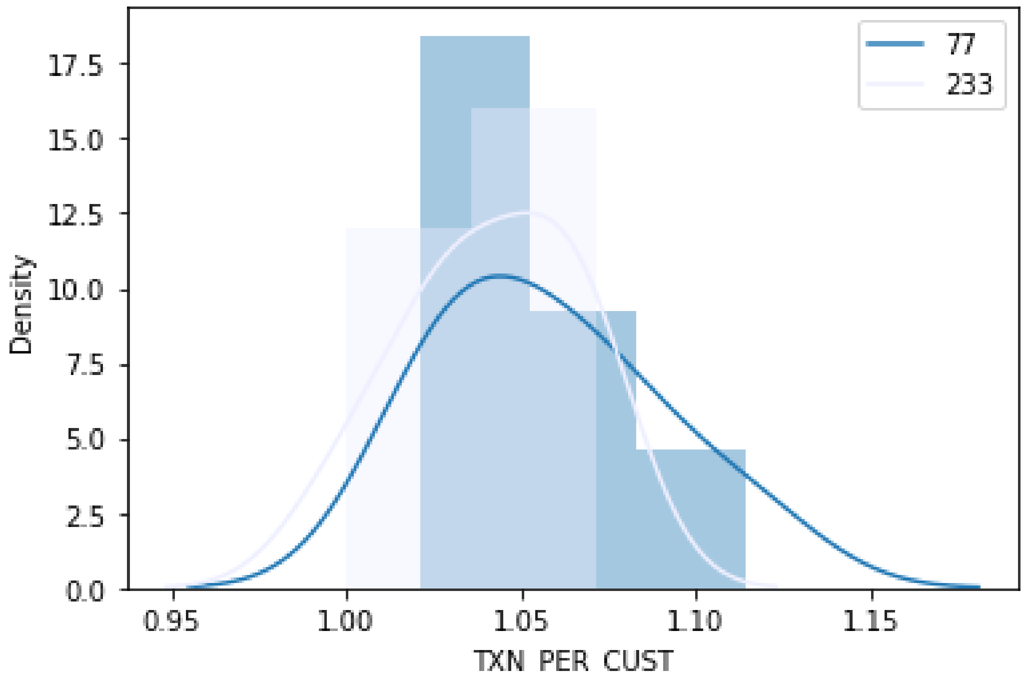
[65]:
```
sns.distplot(metrics.loc[77]['CUSTOMERS'], color='#746ab0')
sns.distplot(metrics.loc[233]['CUSTOMERS'], color='#fbb75c')
plt.legend(labels=['77','233'])
```

[65]: <matplotlib.legend.Legend at 0xce25550>



[59]:
```
sns.distplot(metrics.loc[77]['TXN_PER_CUST'], color=None)
sns.distplot(metrics.loc[233]['TXN_PER_CUST'], color='#eeefff')
plt.legend(labels=['77','233'])
```

[59]: <matplotlib.legend.Legend at 0x56b5fa0>



Now let's import scipy statistics library to do some tests between those two samples 233 and 71

- We will use **ttest_ind** to Calculate the T-test for the means of two independent samples of scores.
- We will also use **ks_2samp** to Performs the two-sample Kolmogorov-Smirnov test for goodness of fit.
- We will also use **t** which is a student's t continuous random variable.

[67]:
```
from scipy.stats import ks_2samp,ttest_ind,t
```

[68]:
```
# difference between control [233] and trial_stores [77] sales
a=[]
for x in metrics.columns:
    a.append(ks_2samp(metrics.loc[77][x], metrics.loc[233][x]))
a=pd.DataFrame(a,index=metrics.columns)
a
```

[68]:

	statistic	pvalue
CUSTOMERS	0.142857	0.999961
PROD_QTY	0.285714	0.962704
TOT_SALES	0.285714	0.962704
PRICE_PER_UNIT	0.285714	0.962704
CHIP_PER_TXN	0.285714	0.962704
TXN_PER_CUST	0.428571	0.575175

All of the p-values are high (say more than 0.05), we can't reject the null hypothesis which we pretend that the null hypothesis be that both stores 77 ans 233 have no difference.

The trial period goes from the start of February 2019 to April 2019. We now want to see if there has been an uplift in overall chip sales.

[71]:
```
b=[]
for x in trial_stores.columns:
    b.append(ttest_ind(trial_stores.loc[77][x].tail(2), trial_stores.
 ↪loc[233][x].tail(2)))
b=pd.DataFrame(b,index=metrics.columns)
b
```

[71]:

	statistic	pvalue
CUSTOMERS	2.586131	0.122618
PROD_QTY	4.043680	0.056063
TOT_SALES	4.267336	0.050769
PRICE_PER_UNIT	-0.634173	0.590828
CHIP_PER_TXN	1.785126	0.216165
TXN_PER_CUST	0.332434	0.771171

[72]:
```
# Now let's calculate critical value which the value start changing from high␣
 ↪to low
t.ppf(0.95,df=7)
```

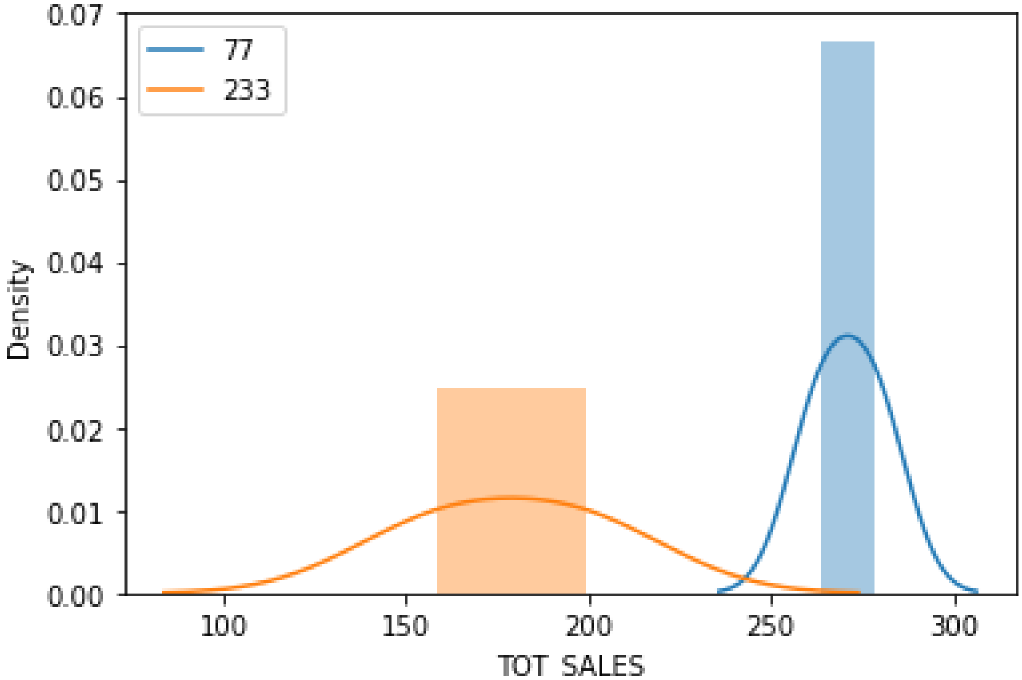
[72]: 1.894578605061305

Since all of the p-values are high (say more than 0.05), we reject the null hypothesis i.e. there means are significantly different.

We can observe that the t-value is much larger than the 95th percentile value of the t-distribution for March and April - i.e. the increase in sales in the trial store in March and April is statistically greater than in the control store.

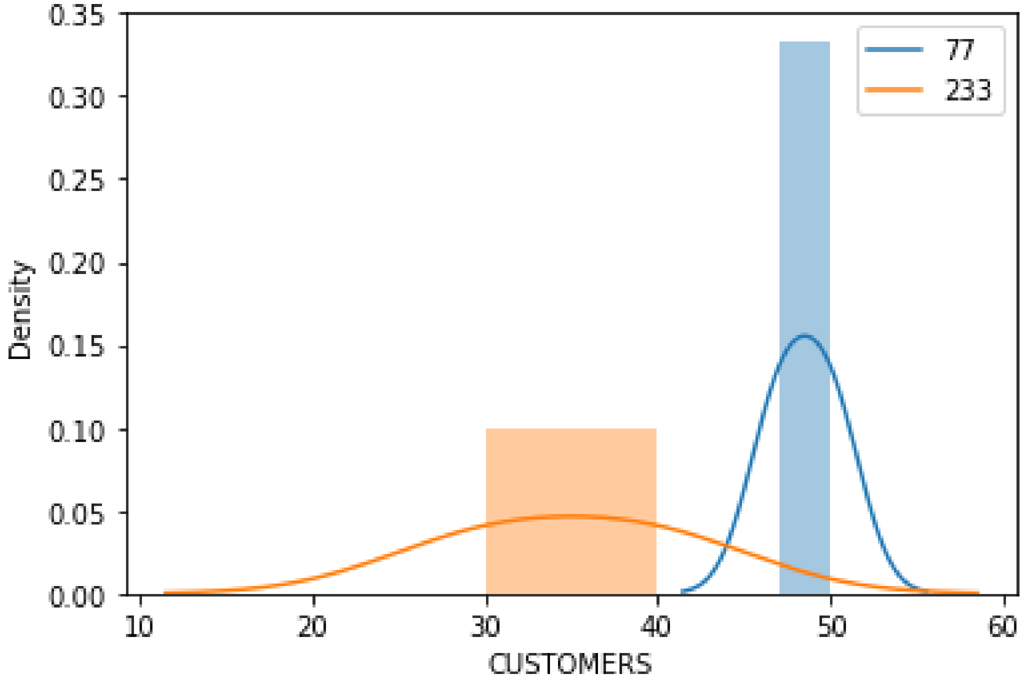
[73]:
```
#calculate means
sns.distplot(trial_stores.loc[77]['TOT_SALES'].tail(2))
sns.distplot(trial_stores.loc[233]['TOT_SALES'].tail(2))
plt.legend(labels=['77','233'])
```

[73]: <matplotlib.legend.Legend at 0xdf69af0>



[75]:
```
sns.distplot(trial_stores.loc[77]['CUSTOMERS'].tail(2))
sns.distplot(trial_stores.loc[233]['CUSTOMERS'].tail(2))
plt.legend(labels=['77','233'])
```

[75]: <matplotlib.legend.Legend at 0xdfaf5b0>



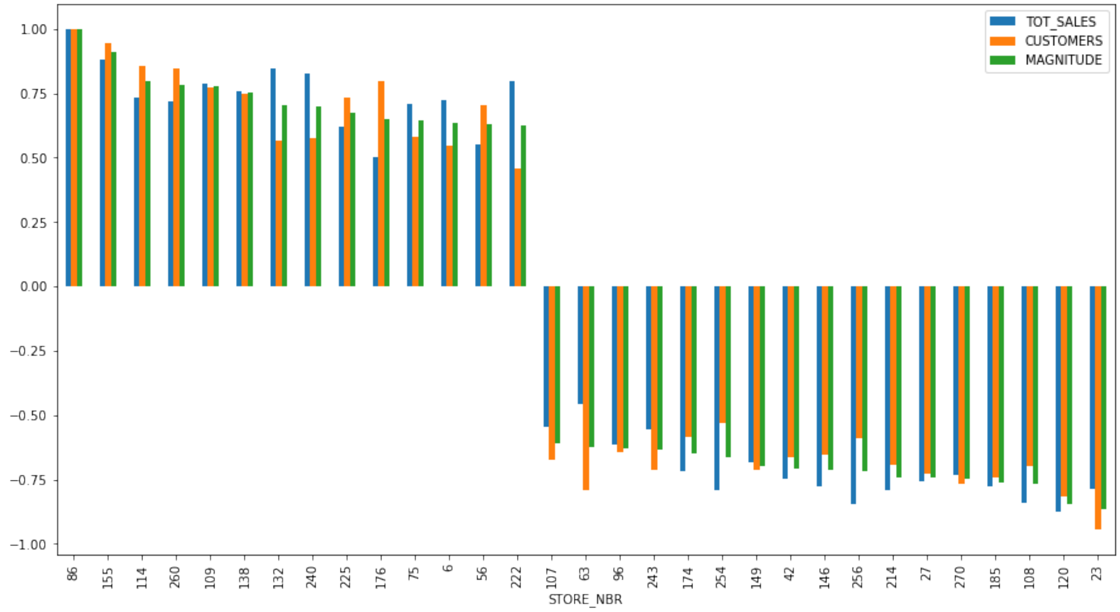
It can be visualized that the is a significant difference in the means, so trial store behavior(77) is different from control store (233).

In other words, The results show that the trial in store 77 is significantly different to its control store in the trial period as the trial store performance lies outside the 5% to 95% confidence interval of the control store in two of the three trial months.

We will repeat those steps for stores 86 and 88

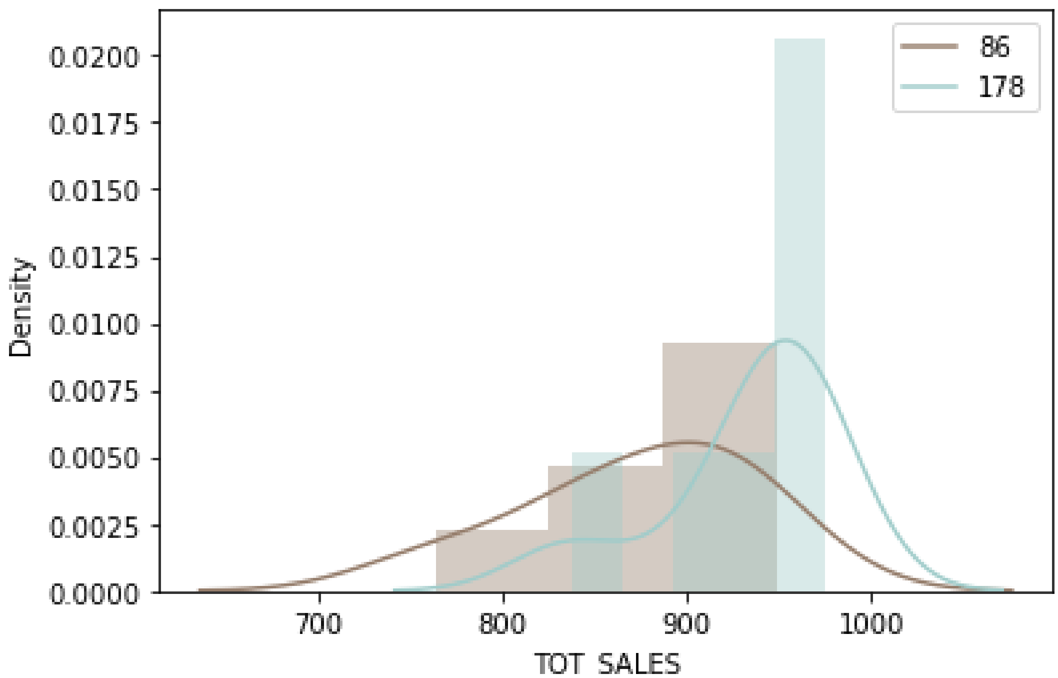
[85]:
```
#Taking 0.6 as threshold correlation
calc_corr_86[(calc_corr_86.MAGNITUDE.abs()>0.6)].plot(kind='bar',figsize=(15,8))
```

[85]:
```
<AxesSubplot:xlabel='STORE_NBR'>
```



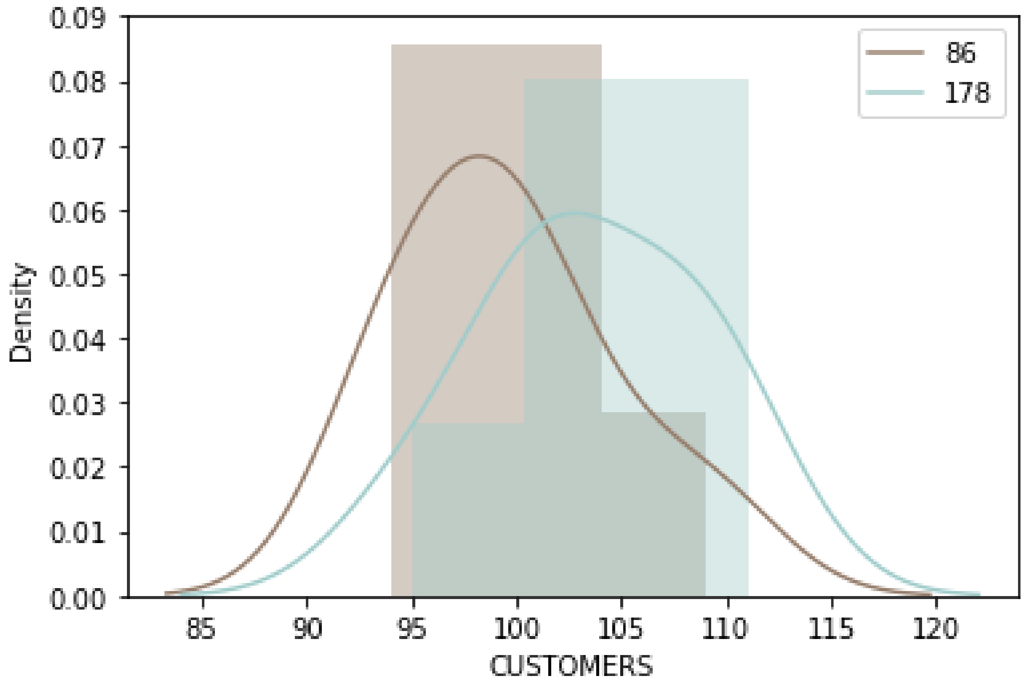
[87]:
```
sns.distplot(metrics.loc[86]['TOT_SALES'], color='#957d6b')
sns.distplot(metrics.loc[178]['TOT_SALES'], color='#a0ccca')
plt.legend(labels=['86','178'])
```

[87]: <matplotlib.legend.Legend at 0x12dbd1f0>



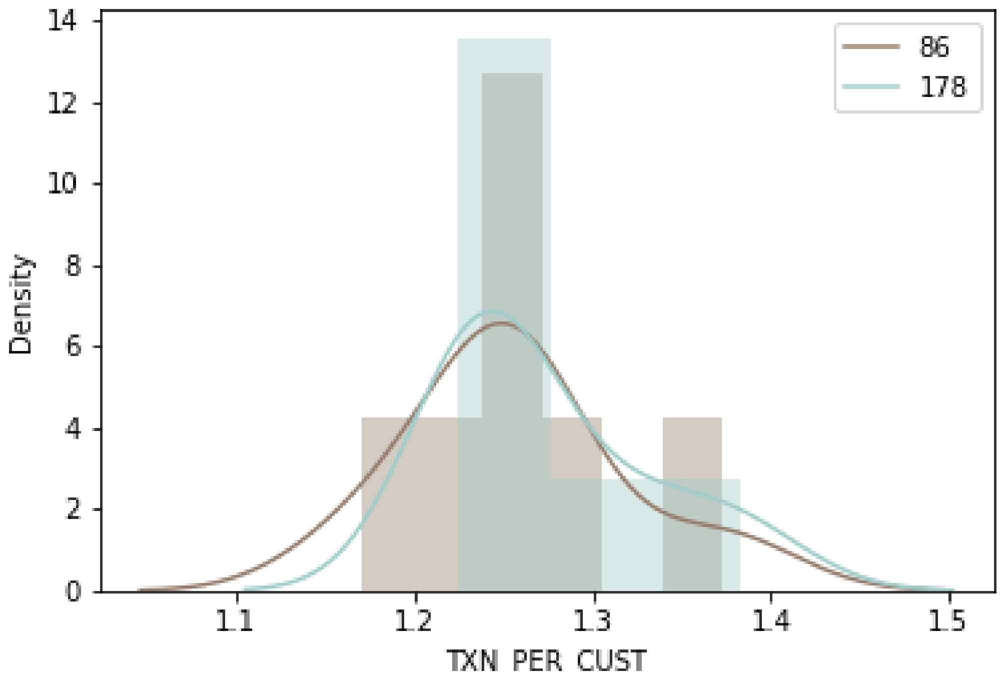
[88]:
```
sns.distplot(metrics.loc[86]['CUSTOMERS'], color='#957d6b')
sns.distplot(metrics.loc[178]['CUSTOMERS'], color='#a0ccca')
plt.legend(labels=['86','178'])
```

[88]: <matplotlib.legend.Legend at 0x12e18b80>



[89]:
```
sns.distplot(metrics.loc[86]['TXN_PER_CUST'], color='#957d6b')
sns.distplot(metrics.loc[178]['TXN_PER_CUST'], color='#a0ccca')
plt.legend(labels=['86','178'])
```

[89]: <matplotlib.legend.Legend at 0x12df6580>



[90]:
```
# difference between control [178] and trial_stores [86] sales
a=[]
for x in metrics.columns:
    a.append(ks_2samp(metrics.loc[86][x], metrics.loc[178][x]))
a=pd.DataFrame(a,index=metrics.columns)
a
```

[90]:

	statistic	pvalue
CUSTOMERS	0.571429	0.212121
PROD_QTY	0.571429	0.212121
TOT_SALES	0.571429	0.212121
PRICE_PER_UNIT	0.285714	0.962704
CHIP_PER_TXN	0.428571	0.575175
TXN_PER_CUST	0.142857	0.999961

[91]:
```
b=[]
for x in trial_stores.columns:
    b.append(ttest_ind(trial_stores.loc[86][x].tail(2), trial_stores.
 ↪loc[178][x].tail(2)))
b=pd.DataFrame(b,index=metrics.columns)
b
```

[91]:

	statistic	pvalue
CUSTOMERS	-1.053609	0.402562

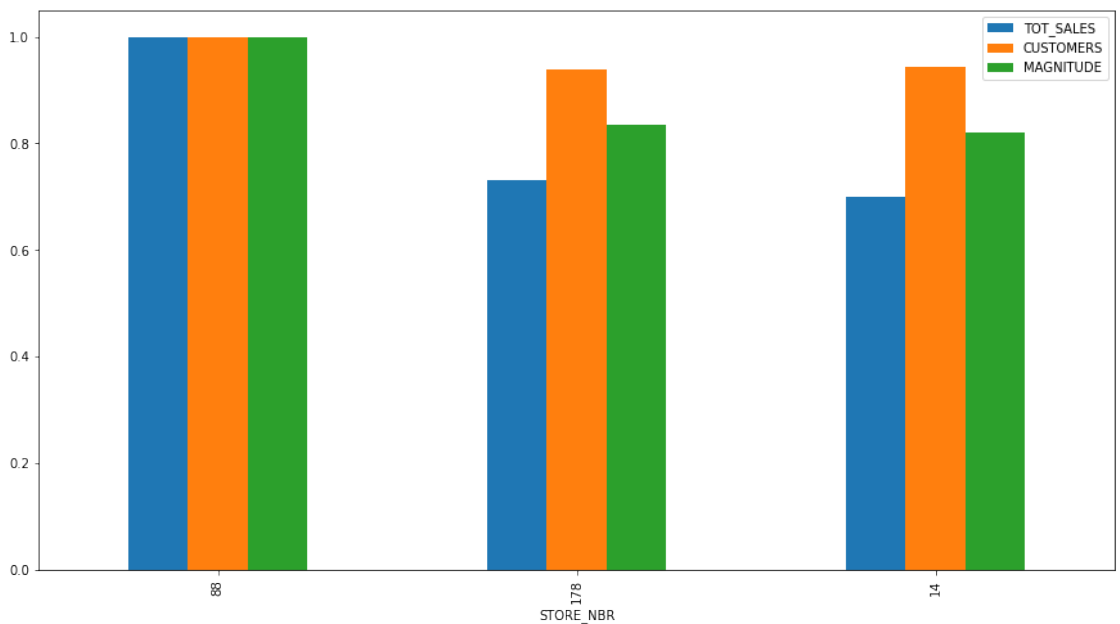
```
PROD_QTY         -1.449893  0.284140
TOT_SALES        -0.972819  0.433255
PRICE_PER_UNIT   -0.251244  0.825083
CHIP_PER_TXN      0.583273  0.618719
TXN_PER_CUST     -5.009394  0.037616
```

[92]:
```
# same critical value will be applied
# Now let's calculate critical value which the value start changing from high␣
↪to low
t.ppf(0.95,df=7)
```

[92]: 1.894578605061305

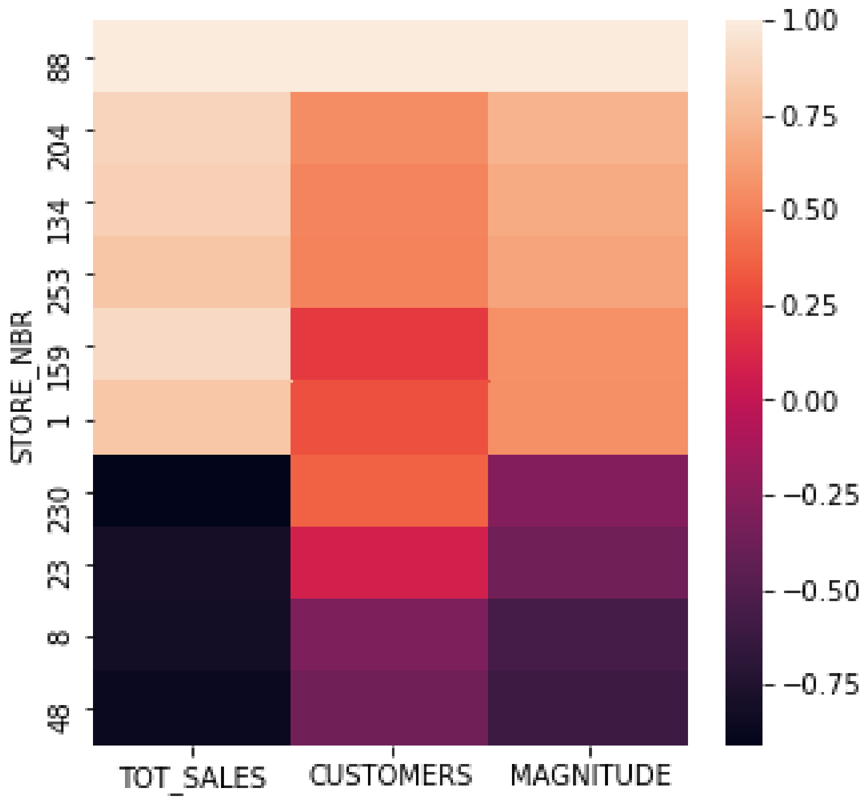
[93]:
```
calc_corr_88[(calc_corr_88.MAGNITUDE.abs()>0.8)].plot(kind='bar',figsize=(15,8))
```

[93]: <AxesSubplot:xlabel='STORE_NBR'>



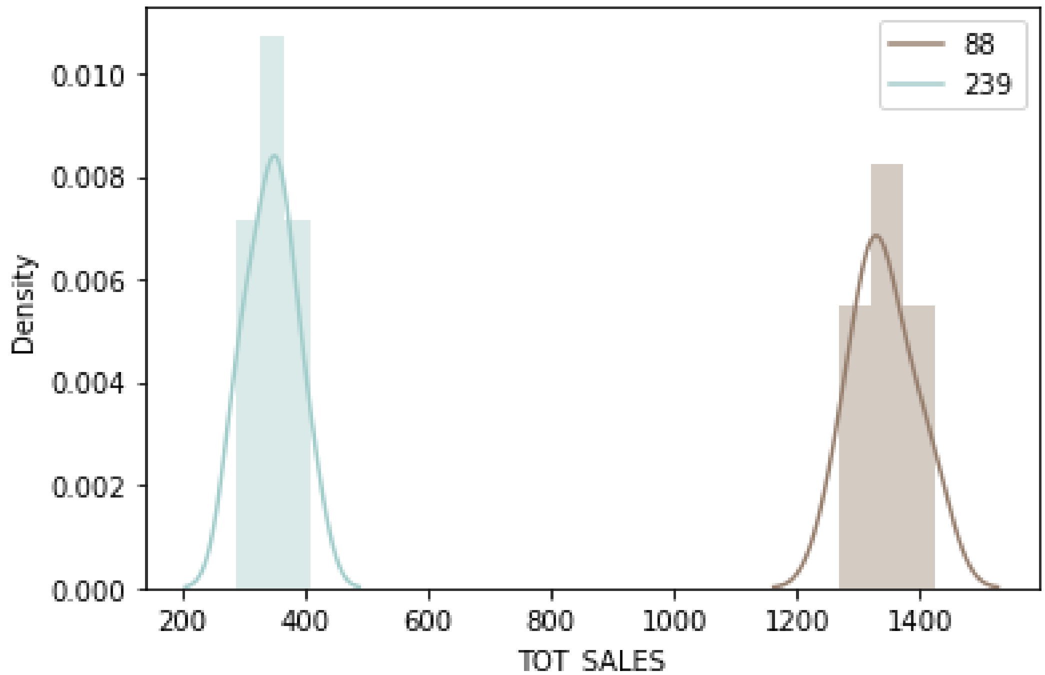
[94]:
```
plt.figure(figsize=(5,5))
sns.heatmap(calc_corr_88[calc_corr_88.TOT_SALES.abs()>0.8])
```

[94]: <AxesSubplot:ylabel='STORE_NBR'>



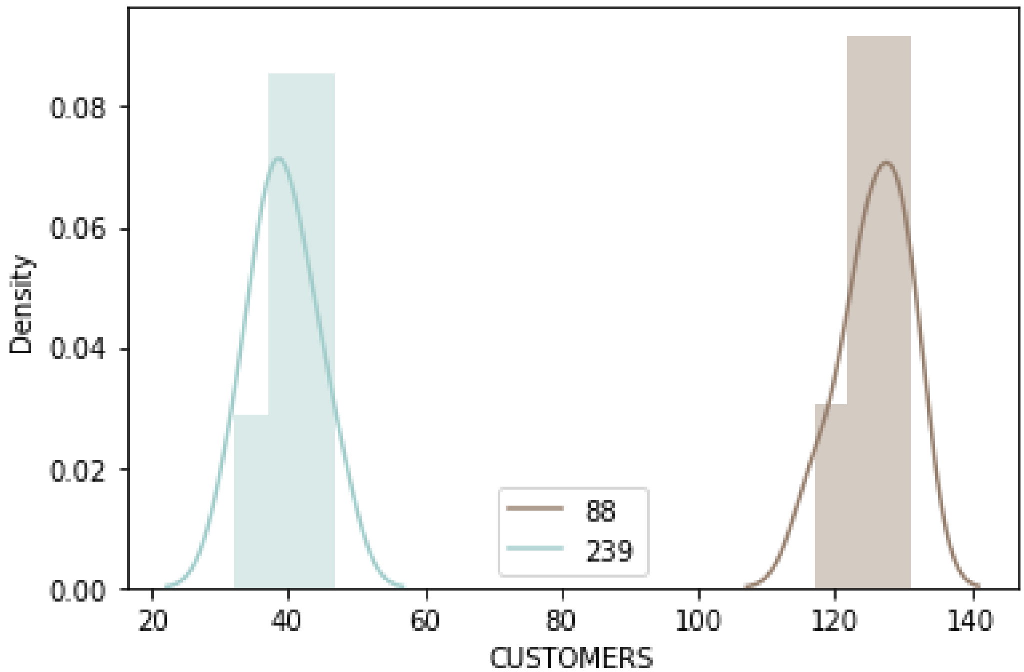
[95]:
```
sns.distplot(metrics.loc[88]['TOT_SALES'], color='#957d6b')
sns.distplot(metrics.loc[239]['TOT_SALES'], color='#a0ccca')
plt.legend(labels=['88','239'])
```

[95]:
```
<matplotlib.legend.Legend at 0x137da760>
```



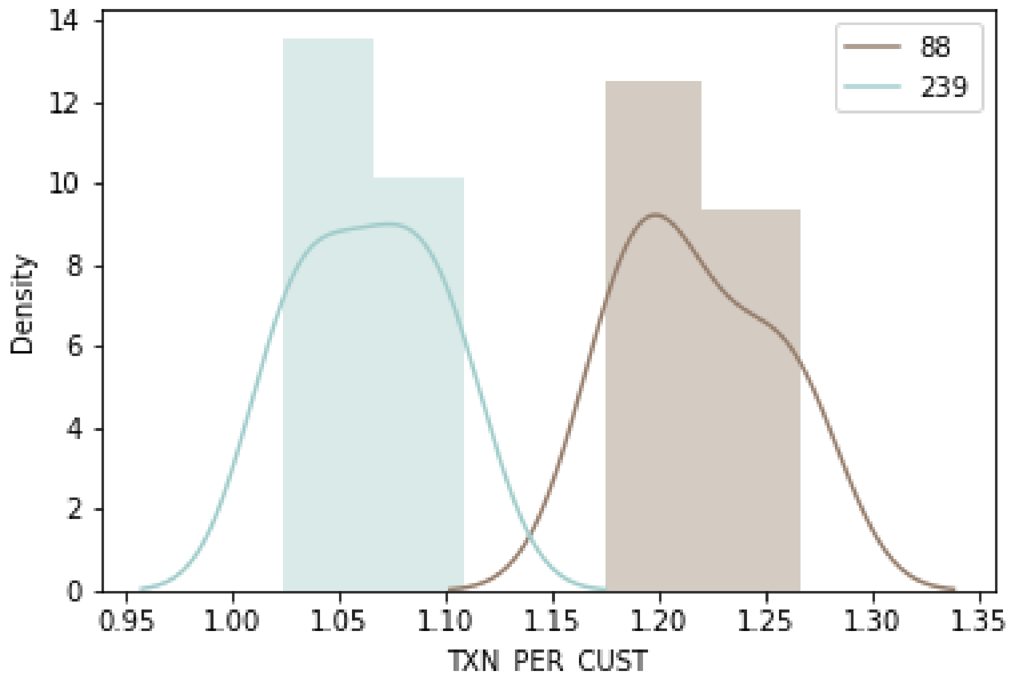

```
[96]: sns.distplot(metrics.loc[88]['CUSTOMERS'], color='#957d6b')
      sns.distplot(metrics.loc[239]['CUSTOMERS'], color='#a0ccca')
      plt.legend(labels=['88','239'])
```

```
[96]: <matplotlib.legend.Legend at 0x137fd880>
```




```
[97]: sns.distplot(metrics.loc[88]['TXN_PER_CUST'], color='#957d6b')
      sns.distplot(metrics.loc[239]['TXN_PER_CUST'], color='#a0ccca')
      plt.legend(labels=['88','239'])
```

```
[97]: <matplotlib.legend.Legend at 0x137f3f10>
```



[98]:
```python
# difference between control [239] and trial_stores [88] sales
a=[]
for x in metrics.columns:
    a.append(ks_2samp(metrics.loc[88][x], metrics.loc[239][x]))
a=pd.DataFrame(a,index=metrics.columns)
a
```

[98]:

	statistic	pvalue
CUSTOMERS	1.000000	0.000583
PROD_QTY	1.000000	0.000583
TOT_SALES	1.000000	0.000583
PRICE_PER_UNIT	0.428571	0.575175
CHIP_PER_TXN	0.857143	0.008159
TXN_PER_CUST	1.000000	0.000583

[99]:
```python
b=[]
for x in trial_stores.columns:
    b.append(ttest_ind(trial_stores.loc[88][x].tail(2), trial_stores.
 ↪loc[239][x].tail(2)))
b=pd.DataFrame(b,index=metrics.columns)
b
```

[99]:

	statistic	pvalue
CUSTOMERS	17.657956	0.003192

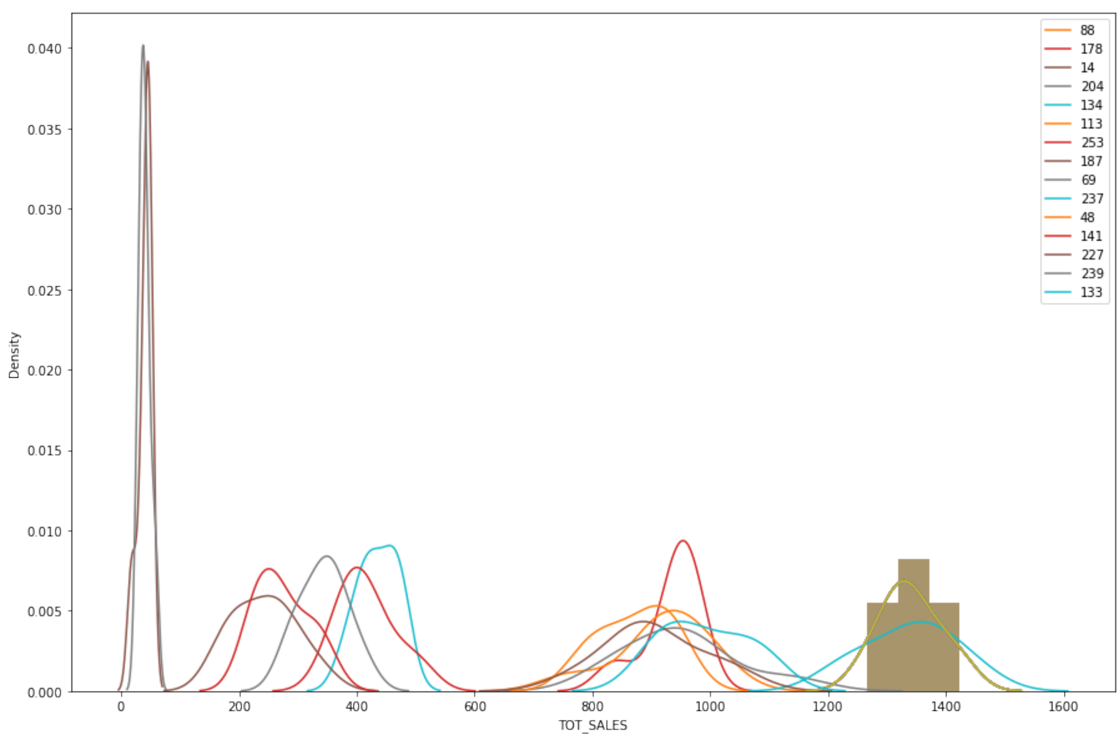
```
PROD_QTY          20.037304  0.002481
TOT_SALES         27.981414  0.001275
PRICE_PER_UNIT    -0.836391  0.490946
CHIP_PER_TXN       1.993648  0.184371
TXN_PER_CUST     118.894737  0.000071
```

[100]:
```python
# same critical value will be applied
# Now let's calculate critical value which the value start changing from high␣
 ↪to low
t.ppf(0.95,df=7)
```

[100]: 1.894578605061305

[79]:
```python
# and so on for the plotting
plt.figure(figsize=(15,10))
for x in calc_corr_88[calc_corr_88.MAGNITUDE.abs()>0.6].index:
    sns.distplot(metrics.loc[88]['TOT_SALES'])
    sns.distplot(metrics.loc[x]['TOT_SALES'],label=x,hist=False)
plt.legend()
```

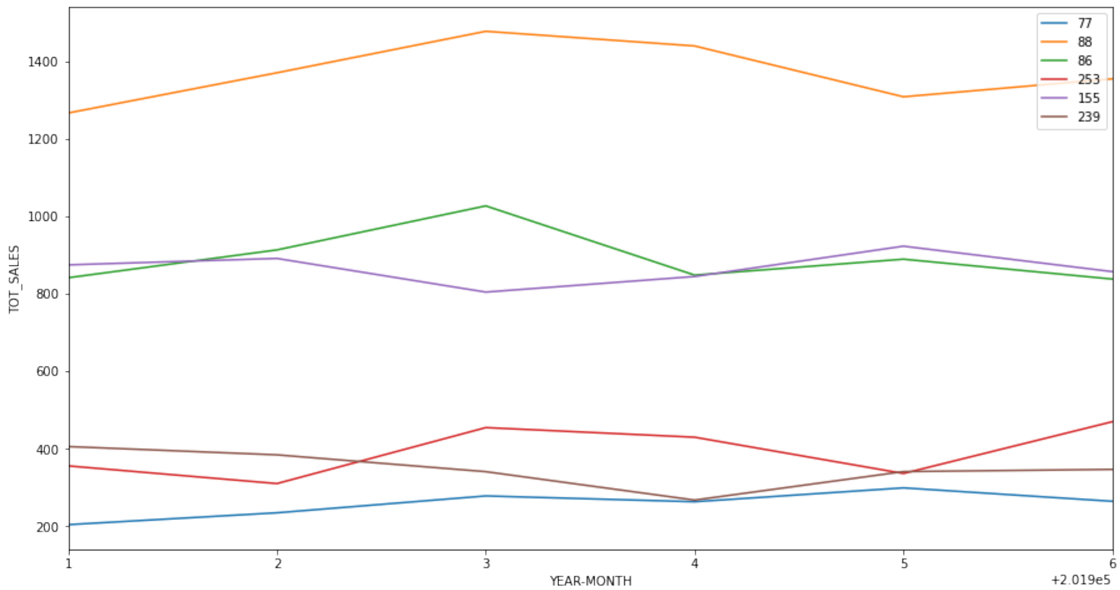
[79]: <matplotlib.legend.Legend at 0xdf067f0>



We will take 239 because it shows different behaviour than the others

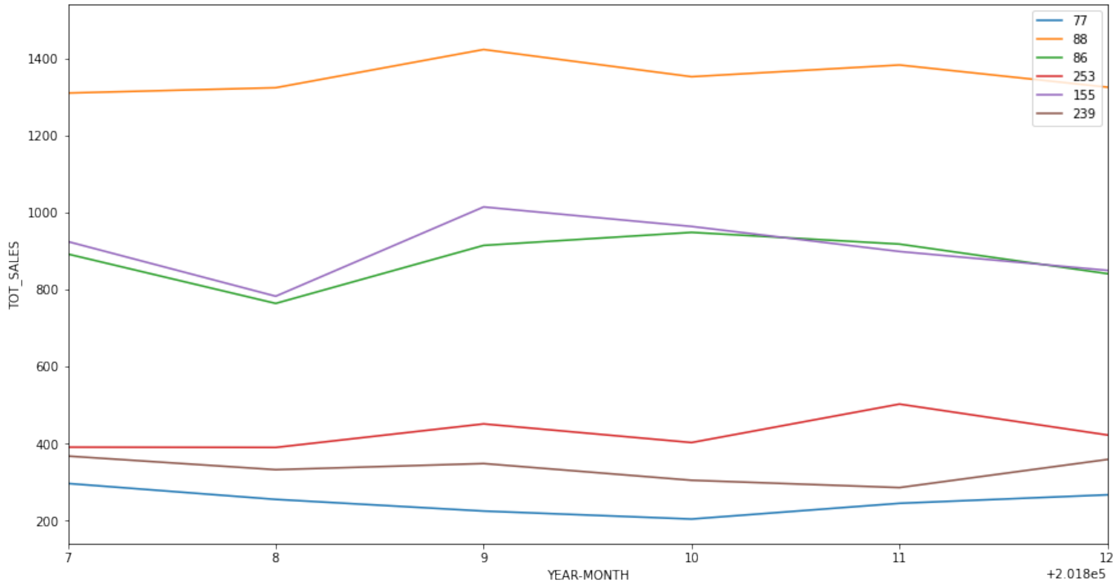
[86]:
```
fig, ax = plt.subplots(figsize=(15, 8))
x=['77','88','86','253','155','239']
for i in  x:
    sns.lineplot(data=mod.loc[int(i)],y='TOT_SALES',x=mod.index.
 ↪get_level_values(1).unique(),label=i)

ax.set_xlim(201901,201906)
```

[86]: (201901.0, 201906.0)



[101]:
```
fig, ax = plt.subplots(figsize=(15, 8))
x=['77','88','86','253','155','239']
for i in  x:
    sns.lineplot(data=mod.loc[int(i)],y='TOT_SALES',x=mod.index.
 ↪get_level_values(1).unique(),label=i)



ax.set_xlim(201807,201812)
```

[101]: (201807.0, 201812.0)



Conclusion

The results for trial stores 77 and 88 during the trial period show a significant difference in at least two of the three trial months but this is not the case for trial store 86. We can check with the client if the implementation of the trial was different in trial store 86 but overall, **The trial shows a significant increase in sales**.

`THANK YOU !!!!`