```
#Importing the libaries + file upload
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
uploaded = files.upload()
```

Choose files Final_chips.csv

**Final_chips.csv**(text/csv) - 28154751 bytes, last modified: 10/09/2025 - 100% done

```
Saving Final_chips.csv to Final_chips.csv
```

```
#import data and quick date to datetime change
df_final_chips = pd.read_csv("Final_chips.csv")
df_final_chips["DATE"] = pd.to_datetime(df_final_chips["DATE"])
df_final_chips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246740 entries, 0 to 246739
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   DATE              246740 non-null  datetime64[ns]
 1   STORE_NBR         246740 non-null  int64
 2   LYLTY_CARD_NBR    246740 non-null  int64
 3   TXN_ID            246740 non-null  int64
 4   PROD_NBR          246740 non-null  int64
 5   PROD_NAME         246740 non-null  object
 6   PROD_QTY          246740 non-null  int64
 7   TOT_SALES         246740 non-null  float64
 8   PACK_SIZE         240676 non-null  float64
 9   BRAND             246740 non-null  object
 10  LIFESTAGE         246740 non-null  object
 11  PREMIUM_CUSTOMER  246740 non-null  object
 12  UNIT_PRICE        246740 non-null  float64
dtypes: datetime64[ns](1), float64(3), int64(5), object(4)
memory usage: 24.5+ MB
```

```
#create month year date column
df_final_chips["MONTH_YEAR"]=df_final_chips["DATE"].dt.strftime("%m/%Y")
df_final_chips
```

| | DATE | STORE_NBR | LYLTY_CARD_NBR | TXN_ID | PROD_NBR | PROD_NAME | PROD_QTY | TOT_SALES | PACK_SIZE | BRAND | LIFE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2018-10-17 | 1 | 1000 | 1 | 5 | Natural Chip Compny SeaSalt175g | 2 | 6.0 | 175.0 | Natural | YO SINGLES/COU |
| **1** | 2019-05-14 | 1 | 1307 | 348 | 66 | CCs Nacho Cheese 175g | 3 | 6.3 | 175.0 | CCs | MII SINGLES/COU |
| **2** | 2019-05-20 | 1 | 1343 | 383 | 61 | Smiths Crinkle Cut Chips Chicken 170g | 2 | 2.9 | 170.0 | Smiths | MII SINGLES/COU |
| **3** | 2018-08-17 | 2 | 2373 | 974 | 69 | Smiths Chip Thinly S/Cream&Onion 175g | 5 | 15.0 | 175.0 | Smiths | MII SINGLES/COU |
| **4** | 2018-08-18 | 2 | 2426 | 1038 | 108 | Kettle Tortilla ChpsHny&Jlpno Chili 150g | 3 | 13.8 | 150.0 | Kettle | MII SINGLES/COU |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **246735** | 2019-03-09 | 272 | 272319 | 270088 | 89 | Kettle Sweet Chilli And Sour Cream 175g | 2 | 10.8 | 175.0 | Kettle | YO SINGLES/COU |
| **246736** | 2018-08-13 | 272 | 272358 | 270154 | 74 | Tostitos Splash Of Lime 175g | 1 | 4.4 | 175.0 | Tostitos | YO SINGLES/COU |
| **246737** | 2018-11-06 | 272 | 272379 | 270187 | 51 | Doritos Mexicana 170g | 2 | 8.8 | 170.0 | Doritos | YO SINGLES/COU |
| **246738** | 2018-12-27 | 272 | 272379 | 270188 | 42 | Doritos Corn Chip Mexican Jalapeno 150g | 2 | 7.8 | 150.0 | Doritos | YO SINGLES/COU |
| **246739** | 2018-09-22 | 272 | 272380 | 270189 | 74 | Tostitos Splash Of Lime 175g | 2 | 8.8 | 175.0 | Tostitos | YO SINGLES/COU |

246740 rows × 14 columns

```
#sort each store and find sales per month

total_grouped = df_final_chips.groupby(["STORE_NBR","MONTH_YEAR"])["TOT_SALES"].sum().reset_index().sort_values(["STORE_NBR",
total_grouped
```

| | STORE_NBR | MONTH_YEAR | TOT_SALES |
|---|---|---|---|
| **0** | 1 | 01/2019 | 149.70 |
| **1** | 1 | 02/2019 | 194.70 |
| **2** | 1 | 03/2019 | 185.20 |
| **3** | 1 | 04/2019 | 177.40 |
| **4** | 1 | 05/2019 | 207.10 |
| **...** | ... | ... | ... |
| **3160** | 272 | 08/2018 | 326.95 |
| **3161** | 272 | 09/2018 | 294.50 |
| **3162** | 272 | 10/2018 | 405.10 |
| **3163** | 272 | 11/2018 | 355.80 |
| **3164** | 272 | 12/2018 | 363.10 |

3165 rows × 3 columns

Next steps: Generate code with `total_grouped` View recommended plots New interactive sheet

```
#now to sum the totals for each store

total_sales_stores = total_grouped.groupby("STORE_NBR")["TOT_SALES"].sum()

total_sales_stores
```

| STORE_NBR | TOT_SALES |
|---|---|
| **1** | 2223.90 |
| **2** | 1854.00 |
| **3** | 12149.65 |
| **4** | 13709.25 |
| **5** | 8802.20 |
| **...** | ... |
| **268** | 2421.85 |
| **269** | 10470.70 |
| **270** | 10519.05 |
| **271** | 8952.30 |
| **272** | 4398.95 |

271 rows × 1 columns

**dtype:** float64

```
#Looking at the specific sales for the trial stores

trial_store = total_sales_stores[[77, 86, 88]]

trial_store
```

| STORE_NBR | TOT_SALES |
|---|---|
| **77** | 2839.20 |
| **86** | 10010.75 |
| **88** | 15445.85 |

**dtype:** float64

```
from pickle import TRUE
```

```
#now to look for a control store, i will start by looking for a control store for store 77

total_sorted = total_sales_stores.sort_values(ascending=TRUE).iloc[70:80]

total_sorted
```

| STORE_NBR | TOT_SALES |
|---|---|
| **205** | 2807.0 |
| **188** | 2823.4 |
| **77** | 2839.2 |
| **141** | 2857.1 |
| **50** | 2859.9 |
| **38** | 2900.4 |
| **90** | 2902.0 |
| **18** | 3021.5 |
| **249** | 3170.6 |
| **256** | 3225.0 |

**dtype:** float64

```
#now to draw out and isolate the 10 stores

stores_control_one = [205,188,77,141,50,38,90,18,249,256]

date_range = pd.date_range("2018-07-01", "2019-01-01", freq="MS").strftime("%m/%Y")

pivot_chips1 = (
    total_grouped[total_grouped['STORE_NBR'].isin(stores_control_one)]
    .pivot_table(
        index="STORE_NBR",
        columns="MONTH_YEAR",
        values="TOT_SALES",
        aggfunc="sum",
        fill_value=0
    )
    .reindex(columns=date_range, fill_value=0)   # ensure the columns follow the desired range
    .reset_index()
)

pivot_chips1
```

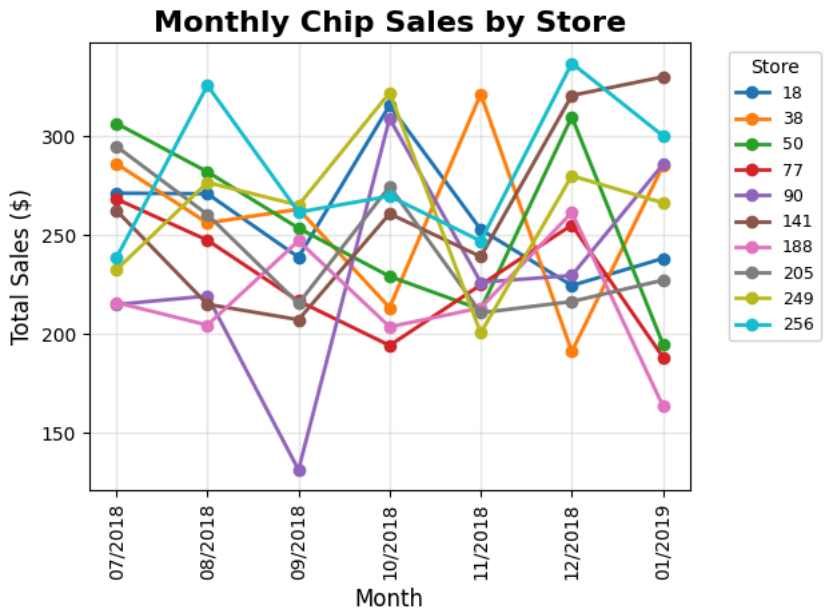
| | STORE_NBR | 07/2018 | 08/2018 | 09/2018 | 10/2018 | 11/2018 | 12/2018 | 01/2019 |
|---|---|---|---|---|---|---|---|---|
| **0** | 18 | 271.3 | 271.0 | 238.9 | 315.7 | 252.9 | 224.6 | 238.3 |
| **1** | 38 | 286.2 | 256.2 | 263.2 | 213.4 | 321.4 | 191.2 | 285.1 |
| **2** | 50 | 306.6 | 282.1 | 253.6 | 229.2 | 212.5 | 309.9 | 195.0 |
| **3** | 77 | 268.4 | 247.5 | 216.8 | 194.3 | 224.9 | 255.2 | 188.4 |
| **4** | 90 | 215.0 | 219.3 | 131.4 | 309.1 | 226.2 | 229.7 | 285.8 |
| **5** | 141 | 262.6 | 215.1 | 207.3 | 260.9 | 239.2 | 320.7 | 330.1 |
| **6** | 188 | 216.0 | 204.6 | 247.2 | 203.7 | 213.3 | 261.5 | 163.7 |
| **7** | 205 | 295.0 | 260.5 | 215.5 | 274.7 | 210.8 | 216.5 | 227.3 |
| **8** | 249 | 232.5 | 276.7 | 265.2 | 322.1 | 200.6 | 280.1 | 266.4 |
| **9** | 256 | 238.5 | 325.8 | 261.7 | 269.8 | 247.1 | 337.0 | 300.3 |

Next steps: Generate code with `pivot_chips1` View recommended plots New interactive sheet

```
#quick plot of stores

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


pivot_chips1.set_index("STORE_NBR").T.plot(kind="line", marker="o", linewidth=2)

plt.title("Monthly Chip Sales by Store", fontsize=16, weight="bold")
plt.xlabel("Month", fontsize=12)
plt.ylabel("Total Sales ($)", fontsize=12)
```

```
plt.xticks(rotation=90, fontsize=10)
plt.yticks(fontsize=10)

plt.legend(title="Store", bbox_to_anchor=(1.05, 1), loc="upper left", fontsize=9)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```




```
#line chart was really messy but 50 looks promising, now i will use the pearson correlation method
#pivot chart is also a bit messy so some cleanup is needed

pivot_chips1.drop(columns="STORE_NBR").T.corr(method="pearson")

pivot_indexed = pivot_chips1.set_index("STORE_NBR")


corr_matrix = pivot_indexed.T.corr(method="pearson")

corr_matrix
```

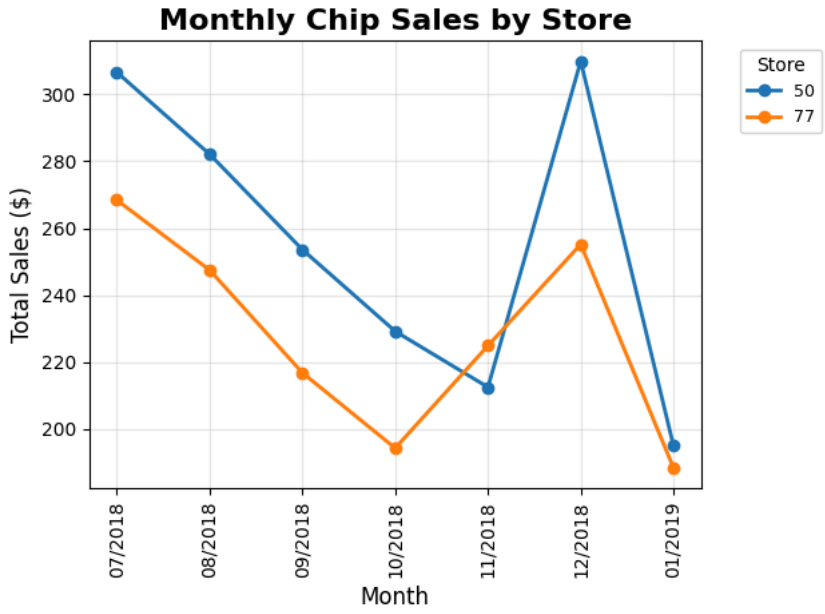
| STORE_NBR | 18 | 38 | 50 | 77 | 90 | 141 | 188 | 205 | 249 | 256 |
|---|---|---|---|---|---|---|---|---|---|---|
| **STORE_NBR** | | | | | | | | | | |
| **18** | 1.000000 | -0.117107 | -0.103438 | -0.188595 | 0.483927 | -0.320535 | -0.331783 | 0.755611 | 0.402487 | -0.333264 |
| **38** | -0.117107 | 1.000000 | -0.444544 | -0.065512 | -0.206960 | -0.284138 | -0.456040 | -0.090743 | -0.827966 | -0.604578 |
| **50** | -0.103438 | -0.444544 | 1.000000 | 0.897701 | -0.396288 | -0.078527 | 0.656892 | 0.352152 | 0.062447 | 0.254287 |
| **77** | -0.188595 | -0.065512 | 0.897701 | 1.000000 | -0.439456 | -0.144593 | 0.542511 | 0.273728 | -0.350070 | 0.086990 |
| **90** | 0.483927 | -0.206960 | -0.396288 | -0.439456 | 1.000000 | 0.590257 | -0.630481 | 0.305958 | 0.396378 | 0.178361 |
| **141** | -0.320535 | -0.284138 | -0.078527 | -0.144593 | 0.590257 | 1.000000 | -0.217203 | -0.125381 | 0.164260 | 0.388209 |
| **188** | -0.331783 | -0.456040 | 0.656892 | 0.542511 | -0.630481 | -0.217203 | 1.000000 | -0.272592 | -0.006157 | 0.089433 |
| **205** | 0.755611 | -0.090743 | 0.352152 | 0.273728 | 0.305958 | -0.125381 | -0.272592 | 1.000000 | 0.255262 | -0.250118 |
| **249** | 0.402487 | -0.827966 | 0.062447 | -0.350070 | 0.396378 | 0.164260 | -0.006157 | 0.255262 | 1.000000 | 0.488774 |
| **256** | -0.333264 | -0.604578 | 0.254287 | 0.086990 | 0.178361 | 0.388209 | 0.089433 | -0.250118 | 0.488774 | 1.000000 |

Next steps: Generate code with corr_matrix | View recommended plots | New interactive sheet

```
#50 has the closest correlation with 77

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

# Filter data for specific stores
chips1_graph = pivot_chips1[pivot_chips1['STORE_NBR'].isin([77, 50])]

# Create the plot
chips1_graph.set_index("STORE_NBR").T.plot(kind="line", marker="o", linewidth=2)
```

```
plt.title("Monthly Chip Sales by Store", fontsize=16, weight="bold")
plt.xlabel("Month", fontsize=12)
plt.ylabel("Total Sales ($)", fontsize=12)
plt.xticks(rotation=90, fontsize=10)
plt.yticks(fontsize=10)

plt.legend(title="Store", bbox_to_anchor=(1.05, 1), loc="upper left", fontsize=9)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```




```
#now to do the same with store 86 looking for the control

total_sorted_two = total_sales_stores.sort_values(ascending=TRUE).iloc[195:205]

total_sorted_two
```

| STORE_NBR | TOT_SALES |
|---|---|
| **247** | 9977.50 |
| **155** | 10004.55 |
| **138** | 10007.80 |
| **86** | 10010.75 |
| **55** | 10063.05 |
| **180** | 10087.80 |
| **160** | 10229.05 |
| **137** | 10230.45 |
| **91** | 10262.00 |
| **114** | 10264.50 |

**dtype:** float64

```
#the same routine with drawing out and isolating the 10 stores

stores_control_two = [247,155,138,86,55,180,160,137,91,114]

date_range = pd.date_range("2018-07-01", "2019-01-01", freq="MS").strftime("%m/%Y")

pivot_chips2 = (
    total_grouped[total_grouped['STORE_NBR'].isin(stores_control_two)]
    .pivot_table(
        index="STORE_NBR",       # one row per store
        columns="MONTH_YEAR",    # each month as a column
        values="TOT_SALES",      # values = total sales
        aggfunc="sum",           # in case there are multiple entries
        fill_value=0
    )
```

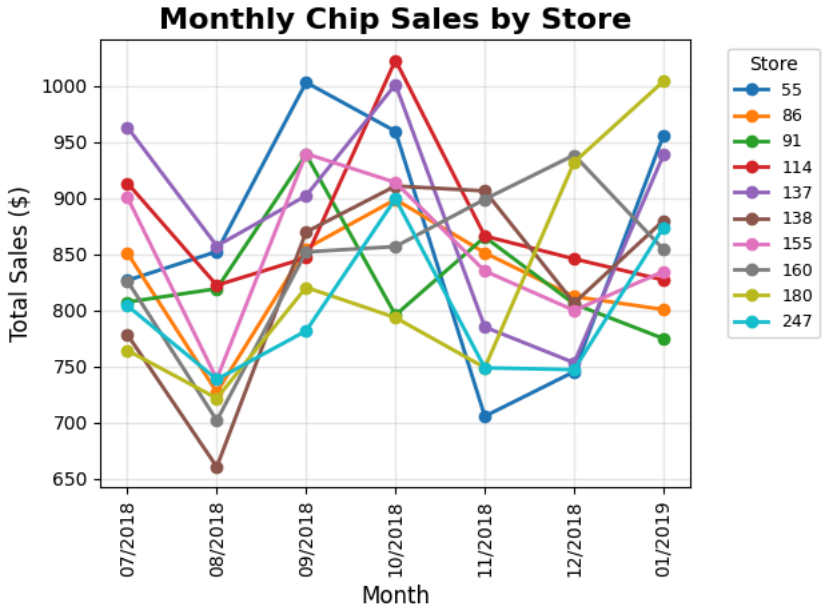
```
    .reindex(columns=date_range, fill_value=0)
    .reset_index()
)

pivot_chips2
```

| | STORE_NBR | 07/2018 | 08/2018 | 09/2018 | 10/2018 | 11/2018 | 12/2018 | 01/2019 |
|---|---|---|---|---|---|---|---|---|
| 0 | 55 | 826.6 | 852.30 | 1003.2 | 959.6 | 705.4 | 745.2 | 956.0 |
| 1 | 86 | 851.0 | 726.85 | 855.0 | 898.8 | 851.2 | 812.2 | 800.6 |
| 2 | 91 | 807.3 | 819.20 | 938.9 | 795.7 | 865.3 | 805.6 | 774.7 |
| 3 | 114 | 913.4 | 822.25 | 846.8 | 1023.0 | 866.2 | 845.8 | 827.0 |
| 4 | 137 | 963.2 | 857.35 | 902.2 | 1001.4 | 785.0 | 753.2 | 938.6 |
| 5 | 138 | 778.2 | 660.30 | 869.8 | 910.8 | 906.4 | 806.4 | 880.0 |
| 6 | 155 | 900.6 | 738.70 | 939.6 | 914.0 | 835.0 | 799.8 | 834.6 |
| 7 | 160 | 826.0 | 701.70 | 852.0 | 856.8 | 899.0 | 938.4 | 854.2 |
| 8 | 180 | 764.2 | 721.40 | 820.4 | 793.4 | 748.8 | 932.0 | 1004.2 |
| 9 | 247 | 804.0 | 738.30 | 781.6 | 899.4 | 748.6 | 747.0 | 874.0 |

Next steps: Generate code with pivot_chips2 View recommended plots New interactive sheet

```
#quick plot of stores

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


pivot_chips2.set_index("STORE_NBR").T.plot(kind="line", marker="o", linewidth=2)

plt.title("Monthly Chip Sales by Store", fontsize=16, weight="bold")
plt.xlabel("Month", fontsize=12)
plt.ylabel("Total Sales ($)", fontsize=12)
plt.xticks(rotation=90, fontsize=10)
plt.yticks(fontsize=10)

plt.legend(title="Store", bbox_to_anchor=(1.05, 1), loc="upper left", fontsize=9)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```




```
#using the same code to get the same correlations
#pivot chart is also a bit messy so some cleanup is needed

pivot_chips2.drop(columns="STORE_NBR").T.corr(method="pearson")

pivot_indexed = pivot_chips2.set_index("STORE_NBR")


corr_matrix = pivot_indexed.T.corr(method="pearson")
```

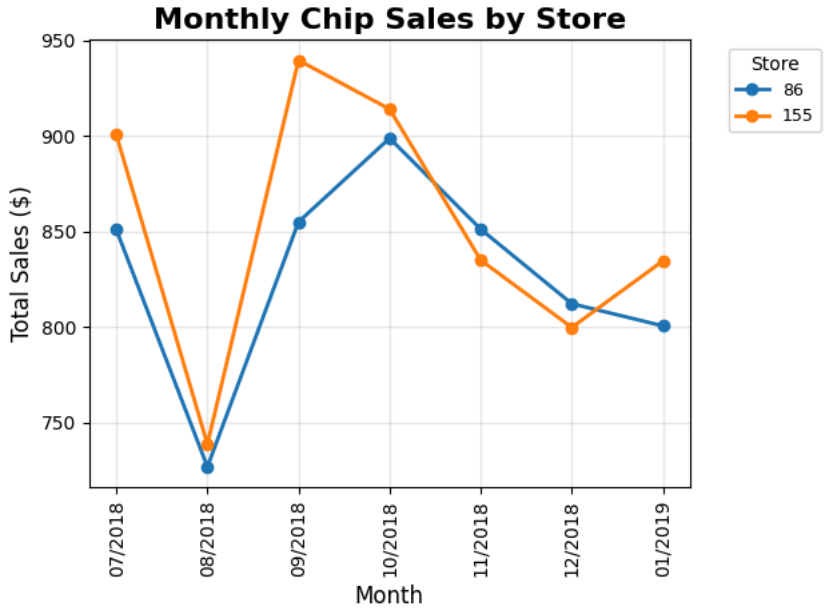
```
corr_matrix
```

| STORE_NBR | 55 | 86 | 91 | 114 | 137 | 138 | 155 | 160 | 180 | 247 |
|---|---|---|---|---|---|---|---|---|---|---|
| **55** | 1.000000 | 0.174236 | 0.133360 | 0.203491 | 0.752363 | 0.203118 | 0.515716 | -0.283672 | 0.220914 | 0.652931 |
| **86** | 0.174236 | 1.000000 | 0.207730 | 0.750924 | 0.382398 | 0.796065 | 0.869532 | 0.600665 | -0.048221 | 0.494357 |
| **91** | 0.133360 | 0.207730 | 1.000000 | -0.227615 | -0.223385 | 0.169317 | 0.388279 | 0.059226 | -0.349533 | -0.439180 |
| **114** | 0.203491 | 0.750924 | -0.227615 | 1.000000 | 0.595630 | 0.403131 | 0.567143 | 0.149464 | -0.282818 | 0.623687 |
| **137** | 0.752363 | 0.382398 | -0.223385 | 0.595630 | 1.000000 | 0.178635 | 0.595705 | -0.324347 | -0.044073 | 0.823460 |
| **138** | 0.203118 | 0.796065 | 0.169317 | 0.403131 | 0.178635 | 1.000000 | 0.635979 | 0.725161 | 0.331896 | 0.534304 |
| **155** | 0.515716 | 0.869532 | 0.388279 | 0.567143 | 0.595705 | 0.635979 | 1.000000 | 0.341222 | -0.022326 | 0.514501 |
| **160** | -0.283672 | 0.600665 | 0.059226 | 0.149464 | -0.324347 | 0.725161 | 0.341222 | 1.000000 | 0.501294 | 0.097111 |
| **180** | 0.220914 | -0.048221 | -0.349533 | -0.282818 | -0.044073 | 0.331896 | -0.022326 | 0.501294 | 1.000000 | 0.363218 |
| **247** | 0.652931 | 0.494357 | -0.439180 | 0.623687 | 0.823460 | 0.534304 | 0.514501 | 0.097111 | 0.363218 | 1.000000 |

Next steps: Generate code with corr_matrix | View recommended plots | New interactive sheet

```
#155 has a decent corrleation with our trial 86

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

# Filter data for specific stores
chips2_graph = pivot_chips2[pivot_chips2['STORE_NBR'].isin([86, 155])]

# Create the plot
chips2_graph.set_index("STORE_NBR").T.plot(kind="line", marker="o", linewidth=2)

plt.title("Monthly Chip Sales by Store", fontsize=16, weight="bold")
plt.xlabel("Month", fontsize=12)
plt.ylabel("Total Sales ($)", fontsize=12)
plt.xticks(rotation=90, fontsize=10)
plt.yticks(fontsize=10)

plt.legend(title="Store", bbox_to_anchor=(1.05, 1), loc="upper left", fontsize=9)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```




```
#now for the final trial store we find a range

total_sorted_three = total_sales_stores.sort_values(ascending=TRUE).iloc[261:271]

total_sorted_three
```

| STORE_NBR | TOT_SALES |
|---|---|
| 26 | 13597.20 |
| 203 | 13623.40 |
| 4 | 13709.25 |
| 199 | 13975.90 |
| 58 | 14256.95 |
| 40 | 14427.30 |
| 237 | 14830.60 |
| 165 | 15188.35 |
| 88 | 15445.85 |
| 226 | 16544.65 |

**dtype:** float64

```
#the same routine with drawing out and isolating the 10 stores

stores_control_three = [26,203,4,199,58,40,237,165,88,226]

date_range = pd.date_range("2018-07-01", "2019-01-01", freq="MS").strftime("%m/%Y")

pivot_chips3 = (
    total_grouped[total_grouped['STORE_NBR'].isin(stores_control_three)]
    .pivot_table(
        index="STORE_NBR",       # one row per store
        columns="MONTH_YEAR",    # each month as a column
        values="TOT_SALES",      # values = total sales
        aggfunc="sum",           # in case there are multiple entries
        fill_value=0
    )
    .reindex(columns=date_range, fill_value=0)
    .reset_index()
)

pivot_chips3
```

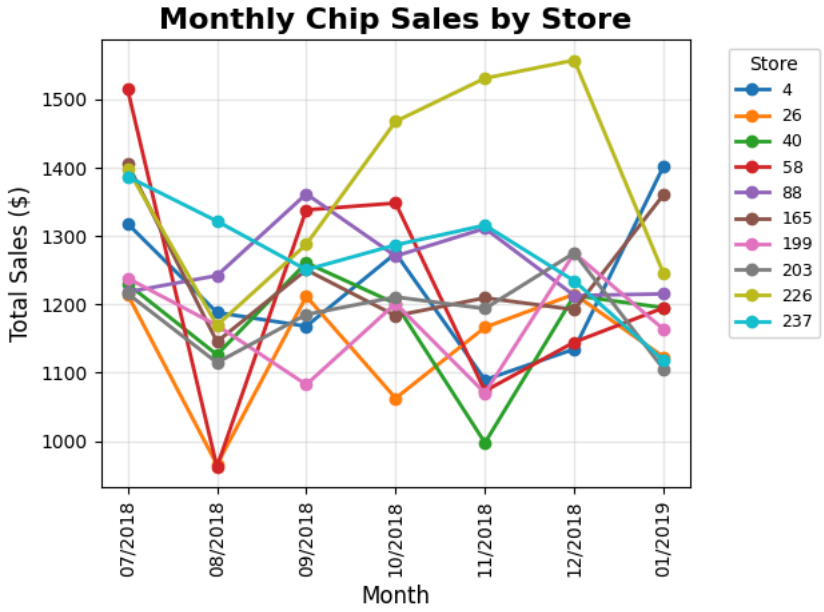
| | STORE_NBR | 07/2018 | 08/2018 | 09/2018 | 10/2018 | 11/2018 | 12/2018 | 01/2019 |
|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 1318.3 | 1188.10 | 1168.0 | 1275.0 | 1089.6 | 1134.6 | 1402.6 |
| 1 | 26 | 1214.4 | 965.30 | 1211.8 | 1062.6 | 1166.4 | 1215.0 | 1123.0 |
| 2 | 40 | 1230.0 | 1126.40 | 1261.0 | 1201.2 | 997.4 | 1213.0 | 1195.2 |
| 3 | 58 | 1515.0 | 962.15 | 1338.4 | 1348.4 | 1073.2 | 1144.8 | 1195.0 |
| 4 | 88 | 1218.2 | 1242.20 | 1361.8 | 1270.8 | 1311.4 | 1213.0 | 1215.4 |
| 5 | 165 | 1406.0 | 1145.40 | 1250.6 | 1183.4 | 1209.6 | 1192.4 | 1360.8 |
| 6 | 199 | 1238.4 | 1169.30 | 1082.6 | 1201.4 | 1070.2 | 1275.2 | 1163.4 |
| 7 | 203 | 1215.8 | 1114.80 | 1185.2 | 1210.9 | 1193.8 | 1275.4 | 1105.2 |
| 8 | 226 | 1398.6 | 1169.25 | 1288.0 | 1468.0 | 1531.2 | 1557.6 | 1245.1 |
| 9 | 237 | 1387.2 | 1321.90 | 1250.8 | 1287.1 | 1316.0 | 1234.4 | 1117.7 |

Next steps: Generate code with pivot_chips3 | View recommended plots | New interactive sheet

```
#quick plot of stores

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


pivot_chips3.set_index("STORE_NBR").T.plot(kind="line", marker="o", linewidth=2)

plt.title("Monthly Chip Sales by Store", fontsize=16, weight="bold")
plt.xlabel("Month", fontsize=12)
plt.ylabel("Total Sales ($)", fontsize=12)
plt.xticks(rotation=90, fontsize=10)
plt.yticks(fontsize=10)

plt.legend(title="Store", bbox_to_anchor=(1.05, 1), loc="upper left", fontsize=9)
plt.grid(alpha=0.3)
```

```
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```




```
#same code for everything,just changing the pivot table

pivot_chips3.drop(columns="STORE_NBR").T.corr(method="pearson")

pivot_indexed = pivot_chips3.set_index("STORE_NBR")


corr_matrix = pivot_indexed.T.corr(method="pearson")

corr_matrix
```

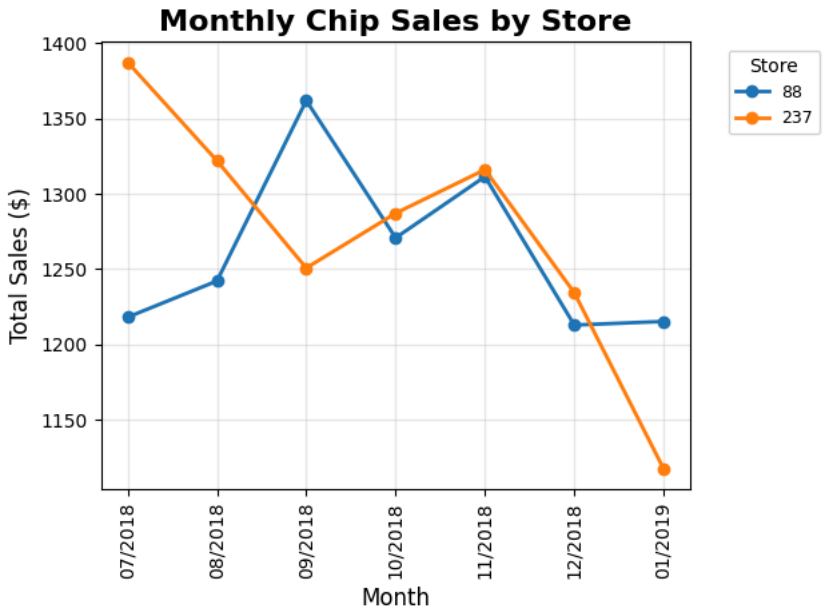
| STORE_NBR | 4 | 26 | 40 | 58 | 88 | 165 | 199 | 203 | 226 | 237 |
|---|---|---|---|---|---|---|---|---|---|---|
| **STORE_NBR** | | | | | | | | | | |
| **4** | 1.000000 | -0.126093 | 0.467968 | 0.471380 | -0.506130 | 0.700157 | 0.313403 | -0.424690 | -0.416095 | -0.336545 |
| **26** | -0.126093 | 1.000000 | 0.289814 | 0.530493 | 0.158180 | 0.502574 | 0.044337 | 0.621314 | 0.523828 | -0.037994 |
| **40** | 0.467968 | 0.289814 | 1.000000 | 0.652823 | -0.139776 | 0.371582 | 0.465831 | 0.192524 | -0.217027 | -0.218527 |
| **58** | 0.471380 | 0.530493 | 0.652823 | 1.000000 | 0.042724 | 0.655980 | 0.244983 | 0.364742 | 0.166934 | 0.215674 |
| **88** | -0.506130 | 0.158180 | -0.139776 | 0.042724 | 1.000000 | -0.289887 | -0.822121 | 0.000674 | 0.000075 | 0.109878 |
| **165** | 0.700157 | 0.502574 | 0.371582 | 0.655980 | -0.289887 | 1.000000 | 0.132120 | -0.114660 | -0.140533 | -0.099816 |
| **199** | 0.313403 | 0.044337 | 0.465831 | 0.244983 | -0.822121 | 0.132120 | 1.000000 | 0.448794 | 0.246681 | 0.073052 |
| **203** | -0.424690 | 0.621314 | 0.192524 | 0.364742 | 0.000674 | -0.114660 | 0.448794 | 1.000000 | 0.861140 | 0.320209 |
| **226** | -0.416095 | 0.523828 | -0.217027 | 0.166934 | 0.000075 | -0.140533 | 0.246681 | 0.861140 | 1.000000 | 0.205770 |
| **237** | -0.336545 | -0.037994 | -0.218527 | 0.215674 | 0.109878 | -0.099816 | 0.073052 | 0.320209 | 0.205770 | 1.000000 |

Next steps: Generate code with corr_matrix | View recommended plots | New interactive sheet

```
#very low correlation for all across the board so going with highest correlation and most similar sales 237

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

# Filter data for specific stores
chips3_graph = pivot_chips3[pivot_chips3['STORE_NBR'].isin([88, 237])]

# Create the plot
chips3_graph.set_index("STORE_NBR").T.plot(kind="line", marker="o", linewidth=2)

plt.title("Monthly Chip Sales by Store", fontsize=16, weight="bold")
plt.xlabel("Month", fontsize=12)
plt.ylabel("Total Sales ($)", fontsize=12)
plt.xticks(rotation=90, fontsize=10)
plt.yticks(fontsize=10)

plt.legend(title="Store", bbox_to_anchor=(1.05, 1), loc="upper left", fontsize=9)
```

```
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

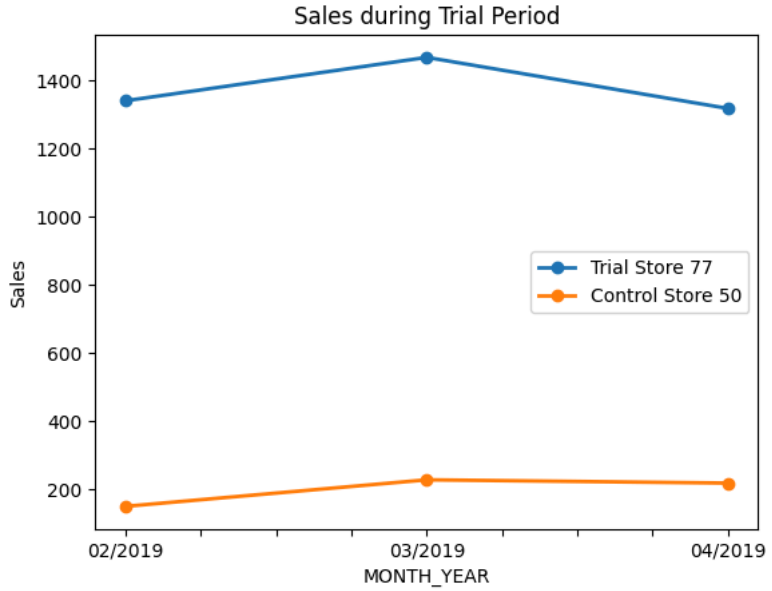
```
<Figure size 1200x600 with 0 Axes>
```




```
#now we can begin to test if total sales are significantly different in the trial period
#creating small dataframes so i can quickly access some checks

trial_period = df_final_chips[(df_final_chips["MONTH_YEAR"] >= "02/2019") & (df_final_chips["MONTH_YEAR"] <= "04/2019")]

t_store77 = trial_period[trial_period["STORE_NBR"] == 77]
c_store50 = trial_period[trial_period["STORE_NBR"] == 50]

t_store86 = trial_period[trial_period["STORE_NBR"] == 86]
c_store155 = trial_period[trial_period["STORE_NBR"] == 155]

t_store88 = trial_period[trial_period["STORE_NBR"] == 88]
c_store237 = trial_period[trial_period["STORE_NBR"] == 237]

t_store77
```

| DATE | STORE_NBR | LYLTY_CARD_NBR | TXN_ID | PROD_NBR | PROD_NAME | PROD_QTY | TOT_SALES | PACK_SIZE | BRAND | L |
|---|---|---|---|---|---|---|---|---|---|---|

```
#checking sales during the trial period for 77 and its control 50

grouped77 = t_store77.groupby("MONTH_YEAR")
grouped50 = c_store50.groupby("MONTH_YEAR")

grouped77["TOT_SALES"].sum().plot(kind="line", marker="o", linewidth=2, label = "Trial Store 77")
grouped50["TOT_SALES"].sum().plot(kind="line", marker="o", linewidth=2, label = "Control Store 50")
plt.ylabel("Sales")
plt.legend()
plt.title("Sales during Trial Period")
plt.show()
```




| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Smiths Crinkle Cut | 2 | 5.8 | 170.0 | Smiths | SINGLES/C |
| | | | | | ... | ... | ... | ... | ... | |
| | | | | | | 1 | 3.0 | 175.0 | Smiths | SINGLES/C |
| | | | | | | 2 | 6.6 | 175.0 | Thins | SINGLES/C |
| | | | | | | 2 | 5.2 | 150.0 | Smith | SINGLES/C |
| | | | | | | 2 | 4.2 | 175.0 | CCs | SINGLES/C |
| | | | | | | 2 | 3.8 | 160.0 | WW | SINGLES/C |

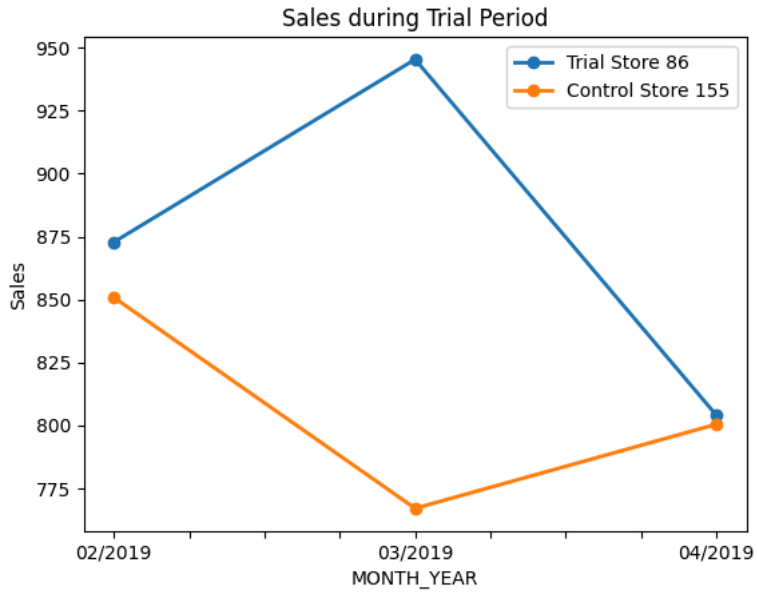
```
#the task also said to check for repeating cusomters so it can be done simply

customer_frequency77 = t_store77["LYLTY_CARD_NBR"].value_counts()

repeating_customers77 = customer_frequency77[customer_frequency77 > 1]

print(f"\nNumber of repeating customers: {len(repeating_customers77)}")
print(f"Total number of unique customers: {len(customer_frequency77)}")
print(f"Percentage of repeating customers: {len(repeating_customers77)/len(customer_frequency77)*100:.2f}%")
```

```
Number of repeating customers: 138
Total number of unique customers: 256
Percentage of repeating customers: 53.91%
```

```
#now we do the same for the control store
customer_frequency50 = c_store50["LYLTY_CARD_NBR"].value_counts()

repeating_customers50 = customer_frequency50[customer_frequency50 > 1]

print(f"\nNumber of repeating customers: {len(repeating_customers50)}")
print(f"Total number of unique customers: {len(customer_frequency50)}")
print(f"Percentage of repeating customers: {len(repeating_customers50)/len(customer_frequency50)*100:.2f}%")
```

```
Number of repeating customers: 7
Total number of unique customers: 112
Percentage of repeating customers: 6.25%
```

```
#checking sales during the trial period for 86 and its control 155

grouped86 = t_store86.groupby("MONTH_YEAR")
grouped155 = c_store155.groupby("MONTH_YEAR")

grouped86["TOT_SALES"].sum().plot(kind="line", marker="o", linewidth=2, label = "Trial Store 86")
grouped155["TOT_SALES"].sum().plot(kind="line", marker="o", linewidth=2, label = "Control Store 155")
plt.ylabel("Sales")
plt.legend()
plt.title("Sales during Trial Period")
plt.show()
```



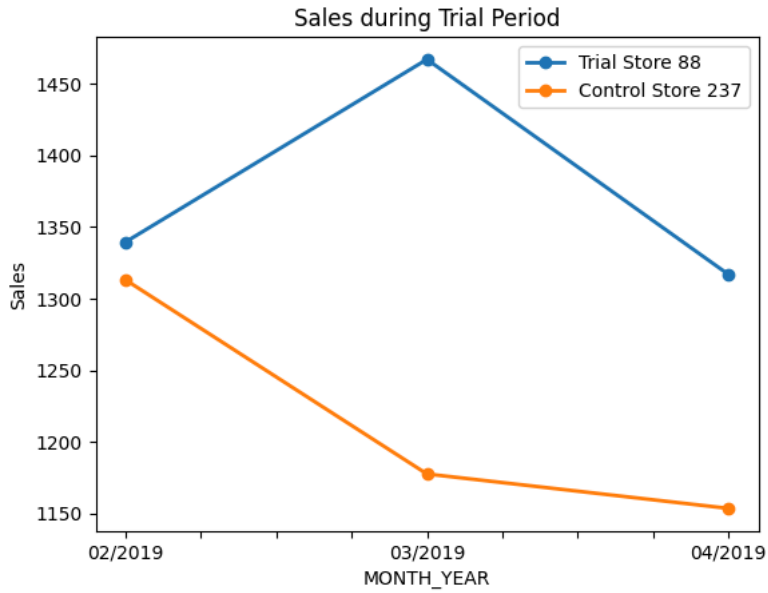

```
#now we do the same repeating customers check for the next trial store

customer_frequency86 = t_store86["LYLTY_CARD_NBR"].value_counts()

repeating_customers86 = customer_frequency86[customer_frequency86 > 1]

print(f"\nNumber of repeating customers: {len(repeating_customers86)}")
print(f"Total number of unique customers: {len(customer_frequency86)}")
print(f"Percentage of repeating customers: {len(repeating_customers86)/len(customer_frequency86)*100:.2f}%")
```

```
Number of repeating customers: 111
Total number of unique customers: 209
Percentage of repeating customers: 53.11%
```

```
#doing he same for the control one

customer_frequency155 = c_store155["LYLTY_CARD_NBR"].value_counts()

repeating_customers155 = customer_frequency155[customer_frequency155 > 1]

print(f"\nNumber of repeating customers: {len(repeating_customers155)}")
print(f"Total number of unique customers: {len(customer_frequency155)}")
print(f"Percentage of repeating customers: {len(repeating_customers155)/len(customer_frequency155)*100:.2f}%")
```

```
Number of repeating customers: 105
Total number of unique customers: 184
Percentage of repeating customers: 57.07%
```

```
#checking sales during the trial period for 88 and its control 237
grouped88 = t_store88.groupby("MONTH_YEAR")
grouped237 = c_store237.groupby("MONTH_YEAR")

grouped88["TOT_SALES"].sum().plot(kind="line", marker="o", linewidth=2, label = "Trial Store 88")
grouped237["TOT_SALES"].sum().plot(kind="line", marker="o", linewidth=2, label = "Control Store 237")
plt.ylabel("Sales")
plt.legend()
plt.title("Sales during Trial Period")
plt.show()
```



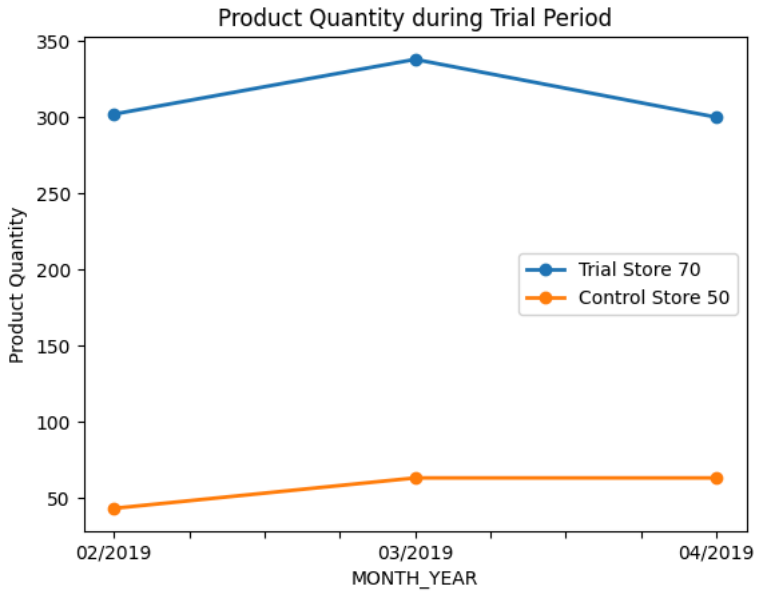

```
#now we do the same repeating customers check for the final trial store

customer_frequency88 = t_store88["LYLTY_CARD_NBR"].value_counts()

repeating_customers88 = customer_frequency88[customer_frequency88 > 1]

print(f"\nNumber of repeating customers: {len(repeating_customers88)}")
print(f"Total number of unique customers: {len(customer_frequency88)}")
print(f"Percentage of repeating customers: {len(repeating_customers88)/len(customer_frequency88)*100:.2f}%")
```

```
Number of repeating customers: 138
Total number of unique customers: 256
Percentage of repeating customers: 53.91%
```

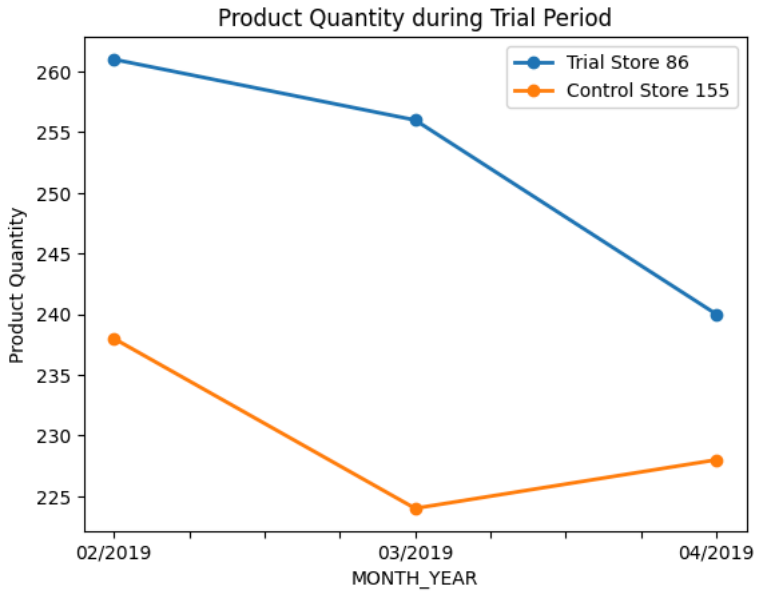
```
#final check for the control tore

customer_frequency237 = c_store237["LYLTY_CARD_NBR"].value_counts()

repeating_customers237 = customer_frequency237[customer_frequency237 > 1]

print(f"\nNumber of repeating customers: {len(repeating_customers237)}")
print(f"Total number of unique customers: {len(customer_frequency237)}")
print(f"Percentage of repeating customers: {len(repeating_customers237)/len(customer_frequency237)*100:.2f}%")
```

```
Number of repeating customers: 106
Total number of unique customers: 251
Percentage of repeating customers: 42.23%
```

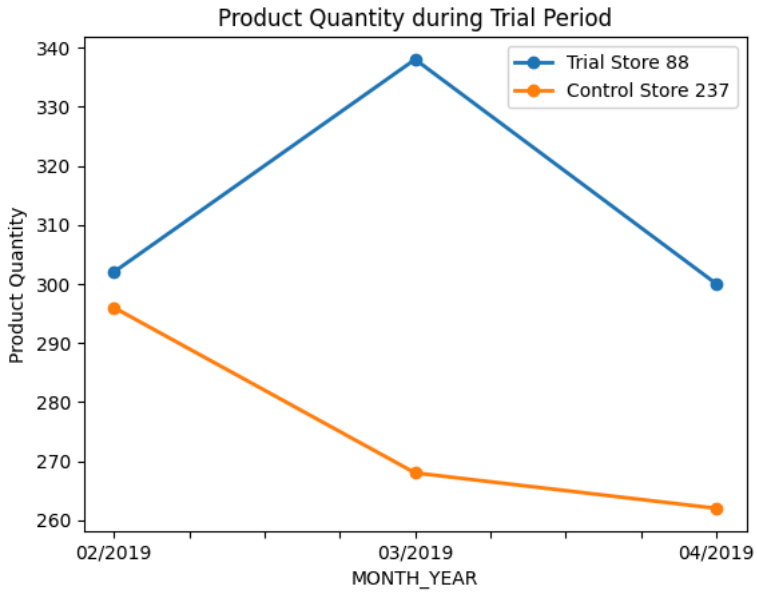
```
#now i want to check for product quanituty during the trial period for each store and control

grouped77["PROD_QTY"].sum().plot(kind="line", marker="o", linewidth=2, label = "Trial Store 70")
grouped50["PROD_QTY"].sum().plot(kind="line", marker="o", linewidth=2, label = "Control Store 50")
plt.ylabel("Product Quantity")
plt.legend()
plt.title("Product Quantity during Trial Period")
plt.show()
```




```
grouped86["PROD_QTY"].sum().plot(kind="line", marker="o", linewidth=2, label = "Trial Store 86")
grouped155["PROD_QTY"].sum().plot(kind="line", marker="o", linewidth=2, label = "Control Store 155")
plt.ylabel("Product Quantity")
plt.legend()
plt.title("Product Quantity during Trial Period")
plt.show()
```




```
grouped88["PROD_QTY"].sum().plot(kind="line", marker="o", linewidth=2, label = "Trial Store 88")
grouped237["PROD_QTY"].sum().plot(kind="line", marker="o", linewidth=2, label = "Control Store 237")
plt.ylabel("Product Quantity")
plt.legend()
plt.title("Product Quantity during Trial Period")
plt.show()
```




```
#everything points towards the trials being sucessful with all trial stores outperforming on sales,product quantity, and repea
#i will now see if it hold true per average customer

grouped77["LYLTY_CARD_NBR"].value_counts().mean()
```

```
np.float64(1.2566844919786095)
```

```
grouped50["LYLTY_CARD_NBR"].value_counts().mean()
```

```
np.float64(1.0347826086956522)
```

```
grouped86["LYLTY_CARD_NBR"].value_counts().mean()
```

```
np.float64(1.2147435897435896)
```

```
grouped155["LYLTY_CARD_NBR"].value_counts().mean()
```

```
np.float64(1.25)
```

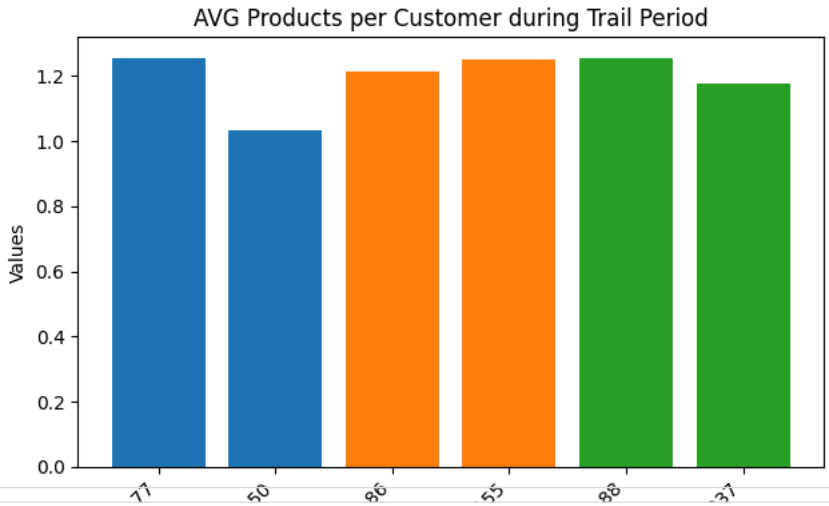
```
grouped88["LYLTY_CARD_NBR"].value_counts().mean()
```

```
np.float64(1.2566844919786095)
```

```
grouped237["LYLTY_CARD_NBR"].value_counts().mean()
```

```
np.float64(1.1766381766381766)
```

```
#now i will plot the values in groups

group1 = ["Trial Store 77", "Control Store 50"]
group2 = ["Trial Store 86", "Control Store 155"]
group3 = ["Trial Store 88", "Control Store 237"]

values_group1 = [1.256,1.034]
values_group2 = [1.214,1.25]
values_group3 = [1.256,1.176]

plt.bar(group1, values_group1, label = group1)
plt.bar(group2, values_group2, label = group2)
plt.bar(group3, values_group3, label = group3)

plt.xticks(rotation=45, ha="right")  # rotate labels so they don't overlap
plt.ylabel("Values")
plt.title("AVG Products per Customer during Trail Period")

plt.tight_layout()  # adjust layout so labels fit nicely
plt.show()
```




```
#Overall I belive that comparing trial stores to control store:
#sales, products sold, amoutn of repeat customers, and average products per customer show
# that during the trial period, trial stores outperform control stores
```