```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv('QVI_data.csv')
```

```
df.shape
```

```
(264834, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264834 entries, 0 to 264833
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   LYLTY_CARD_NBR    264834 non-null  int64
 1   DATE              264834 non-null  object
 2   STORE_NBR         264834 non-null  int64
 3   TXN_ID            264834 non-null  int64
 4   PROD_NBR          264834 non-null  int64
 5   PROD_NAME         264834 non-null  object
 6   PROD_QTY          264834 non-null  int64
 7   TOT_SALES         264834 non-null  float64
 8   PACK_SIZE         264834 non-null  int64
 9   BRAND             264834 non-null  object
 10  LIFESTAGE         264834 non-null  object
 11  PREMIUM_CUSTOMER  264834 non-null  object
dtypes: float64(1), int64(6), object(5)
memory usage: 24.2+ MB
```

```
df.isnull().sum()
```

	0
LYLTY_CARD_NBR	0
DATE	0
STORE_NBR	0
TXN_ID	0
PROD_NBR	0
PROD_NAME	0
PROD_QTY	0
TOT_SALES	0
PACK_SIZE	0
BRAND	0
LIFESTAGE	0
PREMIUM_CUSTOMER	0

dtype: int64

```
df.duplicated().sum()
```

```
np.int64(1)
```

```
df.drop_duplicates(inplace=True)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
df.describe(include = 'all')
```

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND
count	2.648330e+05	264833	264833.000000	2.648330e+05	264833.000000	264833	264833.000000	264833.000000	264833.000000	264833
unique	NaN	364	NaN	NaN	NaN	114	NaN	NaN	NaN	21
top	NaN	2018-12-24	NaN	NaN	NaN	Kettle Mozzarella Basil & Pesto 175g	NaN	NaN	NaN	KETTLE
freq	NaN	939	NaN	NaN	NaN	3304	NaN	NaN	NaN	41288
mean	1.355489e+05	NaN	135.079529	1.351577e+05	56.583598	NaN	1.905812	7.299351	182.425540	NaN
std	8.058003e+04	NaN	76.784189	7.813305e+04	32.826498	NaN	0.343437	2.527244	64.325268	NaN
min	1.000000e+03	NaN	1.000000	1.000000e+00	1.000000	NaN	1.000000	1.500000	70.000000	NaN
25%	7.002100e+04	NaN	70.000000	6.760000e+04	28.000000	NaN	2.000000	5.400000	150.000000	NaN
50%	1.303570e+05	NaN	130.000000	1.351370e+05	56.000000	NaN	2.000000	7.400000	170.000000	NaN

```
df.nunique()
```

	0
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

```
txn_dupes = df[df.duplicated(subset=['TXN_ID'], keep=False)].sort_values('TXN_ID')
```

```
txn_dupes.head(10)
```

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE
500	1446	2019-06-20	1	517	99	Pringles Sthrn FriedChicken 134g	2	7.4	134	PRINGLES	OLDER FAMILIES
501	1446	2019-06-20	1	517	9	Kettle Tortilla ChpsBtroot&Ricotta 150g	2	9.2	150	KETTLE	OLDER FAMILIES
612	2034	2018-12-20	2	628	104	Infuzions Thai SweetChili PotatoMix 110g	1	3.8	110	INFUZIONS	YOUNG SINGLES/COUPLES
611	2034	2018-12-20	2	628	95	Sunbites Whlegrn Crisps Frch/Onin 90g	1	1.7	90	SUNBITES	YOUNG SINGLES/COUPLES
1114	3008	2018-08-26	3	1142	33	Cobs Popd Swt/Chlli &Sr/Cream Chips 110g	2	7.6	110	COBS	RETIREES

```
df_grouped = df.groupby('TXN_ID').agg({
    'DATE': 'first',
    'STORE_NBR': 'first',
    'LYLTY_CARD_NBR': 'first',
```

```
    'PROD_QTY': lambda x: ', '.join(map(str, x)),
    'PROD_NBR': 'nunique',
    'PROD_NAME': lambda x: ', '.join(set(x)),
    'BRAND': lambda x: ', '.join(set(x)),
    'PACK_SIZE': lambda x: ', '.join(map(str,x)),
    'LIFESTAGE': 'first',
    'PREMIUM_CUSTOMER': 'first',
    'TOT_SALES': 'sum'
}).reset_index()
```

```
df_grouped.head()
```

	TXN_ID	DATE	STORE_NBR	LYLTY_CARD_NBR	PROD_QTY	PROD_NBR	PROD_NAME	BRAND	PACK_SIZE	LIFESTAGE	PREMIUM_CUST
0	1	2018-10-17	1	1000	2	1	Natural Chip Compny SeaSalt175g	NATURAL	175	YOUNG SINGLES/COUPLES	Prer
1	2	2018-09-16	1	1002	1	1	Red Rock Deli Chikn&Garlic Aioli 150g	RRD	150	YOUNG SINGLES/COUPLES	Mainst
							Grain Waves				

```
df_grouped['TOTAL_PROD_QTY'] = df_grouped['PROD_QTY'].apply(lambda x: sum(map(int, x.split(', '))))
```

```
df_grouped.shape
```

```
(263125, 13)
```

```
df_grouped.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 263125 entries, 0 to 263124
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   TXN_ID            263125 non-null  int64
 1   DATE              263125 non-null  object
 2   STORE_NBR         263125 non-null  int64
 3   LYLTY_CARD_NBR    263125 non-null  int64
 4   PROD_QTY          263125 non-null  object
 5   PROD_NBR          263125 non-null  int64
 6   PROD_NAME         263125 non-null  object
 7   BRAND             263125 non-null  object
 8   PACK_SIZE         263125 non-null  object
 9   LIFESTAGE         263125 non-null  object
 10  PREMIUM_CUSTOMER  263125 non-null  object
 11  TOT_SALES         263125 non-null  float64
 12  TOTAL_PROD_QTY    263125 non-null  int64
dtypes: float64(1), int64(5), object(7)
memory usage: 26.1+ MB
```

```
df_grouped['DATE'] = pd.to_datetime(df_grouped['DATE'])
```

```
df_grouped.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 263125 entries, 0 to 263124
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   TXN_ID            263125 non-null  int64
 1   DATE              263125 non-null  datetime64[ns]
 2   STORE_NBR         263125 non-null  int64
 3   LYLTY_CARD_NBR    263125 non-null  int64
 4   PROD_QTY          263125 non-null  object
 5   PROD_NBR          263125 non-null  int64
 6   PROD_NAME         263125 non-null  object
 7   BRAND             263125 non-null  object
 8   PACK_SIZE         263125 non-null  object
 9   LIFESTAGE         263125 non-null  object
 10  PREMIUM_CUSTOMER  263125 non-null  object
 11  TOT_SALES         263125 non-null  float64
 12  TOTAL_PROD_QTY    263125 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(5), object(6)
memory usage: 26.1+ MB
```

```
df_grouped['MONTH'] = df_grouped['DATE'].dt.month
df_grouped['YEAR'] = df_grouped['DATE'].dt.year
```

```
df_grouped['MONTH_YEAR'] = df_grouped['DATE'].dt.to_period('M')
```

```
monthly_metrics = df_grouped.groupby(['STORE_NBR','MONTH_YEAR']).agg(
    total_sales = ('TOT_SALES','sum'),
    num_customers= ('LYLTY_CARD_NBR',pd.Series.nunique),
    num_transactions = ('TXN_ID',pd.Series.nunique)
).reset_index()
```

```
monthly_metrics['avg_txn_per_cust'] = monthly_metrics['num_transactions']/monthly_metrics['num_customers']
monthly_metrics.head(10)
```

	STORE_NBR	MONTH_YEAR	total_sales	num_customers	num_transactions	avg_txn_per_cust
0	1	2018-07	206.9	49	52	1.061224
1	1	2018-08	176.1	42	43	1.023810
2	1	2018-09	278.8	59	62	1.050847
3	1	2018-10	188.1	44	45	1.022727
4	1	2018-11	192.6	46	47	1.021739
5	1	2018-12	189.6	42	47	1.119048
6	1	2019-01	154.8	35	36	1.028571
7	1	2019-02	225.4	52	55	1.057692
8	1	2019-03	192.9	45	49	1.088889
9	1	2019-04	192.9	42	43	1.023810

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

Define Function to Compare Similar Stores (Control Store Selection)

```
from scipy.stats import pearsonr
```

```
def find_best_control_store (trial_store , df_metrics , pretrial_months):
  trial_data  = df_metrics[(df_metrics['STORE_NBR'] == trial_store)&
                           (df_metrics['MONTH_YEAR'].isin(pretrial_months))]
  other_stores = df_metrics['STORE_NBR'].unique()
  other_stores = [s for s in other_stores if s != trial_store]

  results = []

  for store in other_stores:
      store_data = df_metrics[(df_metrics['STORE_NBR'] == store)&
                              (df_metrics['MONTH_YEAR'].isin(pretrial_months))]

      if len(store_data) != len(trial_data):
          continue

      corr_sales, _ = pearsonr(trial_data['total_sales'], store_data['total_sales'])
      corr_customers, _ = pearsonr(trial_data['num_customers'], store_data['num_customers'])

      sales_mape = (abs(trial_data['total_sales'].values - store_data['total_sales'].values) / trial_data['total_sales'].values).mean()
      customers_mape = (abs(trial_data['num_customers'].values - store_data['num_customers'].values) / trial_data['num_customers'].values).m

      score = (corr_sales + corr_customers) - (sales_mape + customers_mape)
      results.append((store, score))

  results.sort(key=lambda x: x[1], reverse=True)
  return results[0][0] if results else None
```

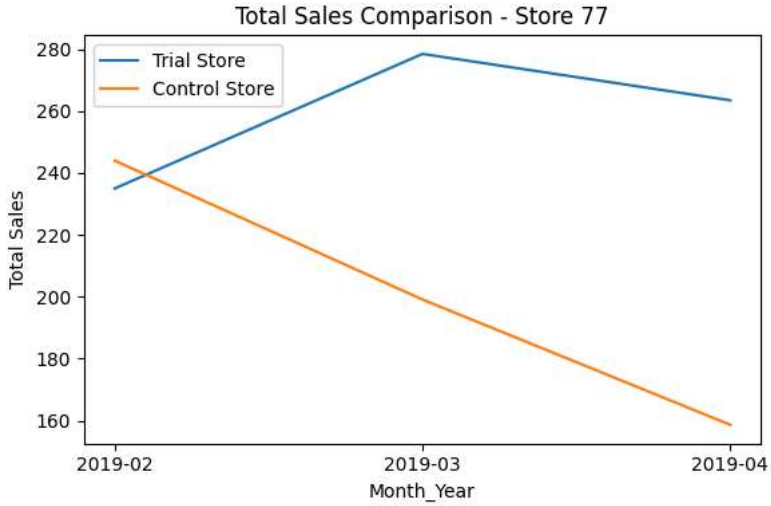
```
pretrial_month =pd.period_range('2018-07', '2019-01',freq = 'M')
trial_stores =[77,86,88]
control_stores = {}
for store in trial_stores:
  control_store = find_best_control_store(store,monthly_metrics,pretrial_month)
  control_stores[store] = control_store

print("Control Stores selected:")
print(control_stores)
```

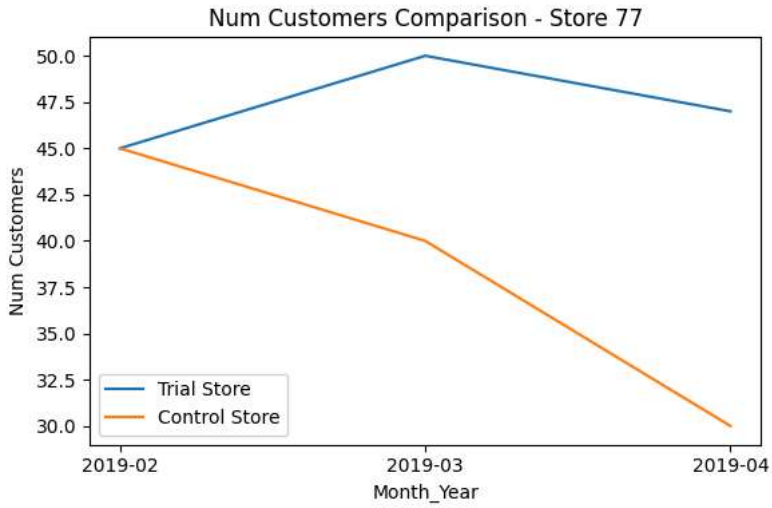
```
Control Stores selected:
{77: np.int64(233), 86: np.int64(155), 88: np.int64(237)}
```

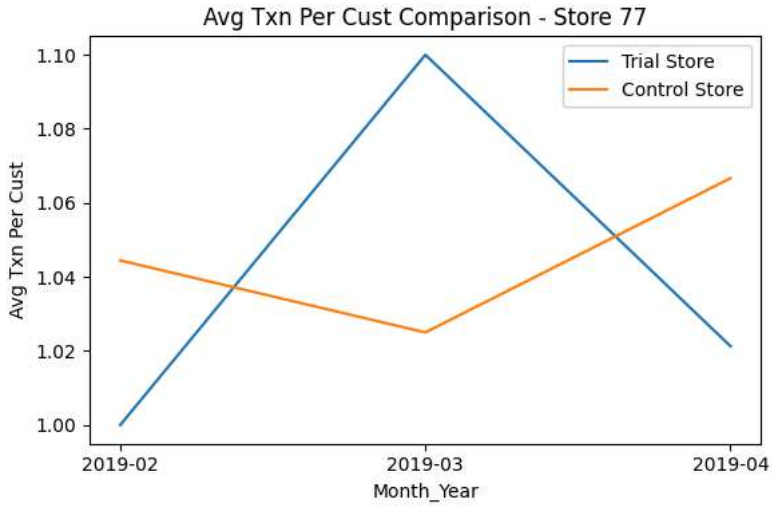
COMPARE TRIAL VS CONTROL STORE IN TRIAL PERIOD

```
import matplotlib.pyplot as plt
trial_period = pd.period_range('2019-02','2019-04',freq = 'M')


for trial_store , control_store in control_stores.items():
    print(f"\n Trial Store :{trial_store},Control Store :{control_store}")

    trial = monthly_metrics[(monthly_metrics['STORE_NBR']== trial_store)&
                            (monthly_metrics['MONTH_YEAR'].isin(trial_period))]

    control = monthly_metrics[(monthly_metrics['STORE_NBR']== control_store)&
                            (monthly_metrics['MONTH_YEAR'].isin(trial_period))]

    merged = trial.merge(control , on ='MONTH_YEAR',suffixes = ('_trial','_control'))

    for metric in ['total_sales', 'num_customers','avg_txn_per_cust']:
        plt.figure(figsize = (6,4))
        plt.plot(merged['MONTH_YEAR'].astype(str),merged[f'{metric}_trial'],label = 'Trial Store')
        plt.plot(merged['MONTH_YEAR'].astype(str),merged[f'{metric}_control'],label = 'Control Store')
        plt.title(f'{metric.replace("_"," ").title()} Comparison - Store {trial_store}')
        plt.ylabel(metric.replace("_"," ").title())
        plt.xlabel('Month_Year')
        plt.legend()
        plt.tight_layout()
        plt.show()
```

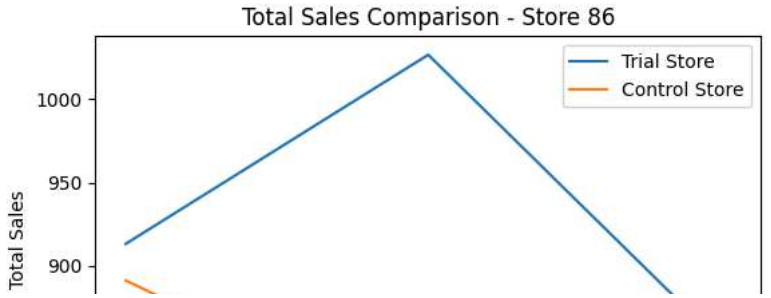
```
Trial Store :77,Control Store :233
```

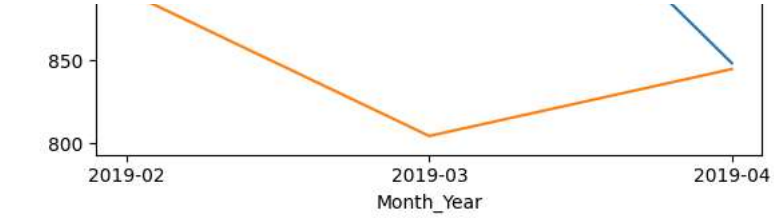


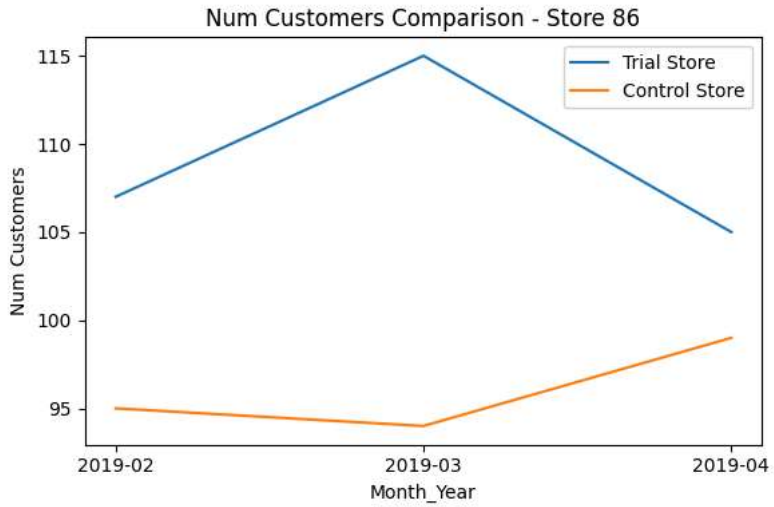


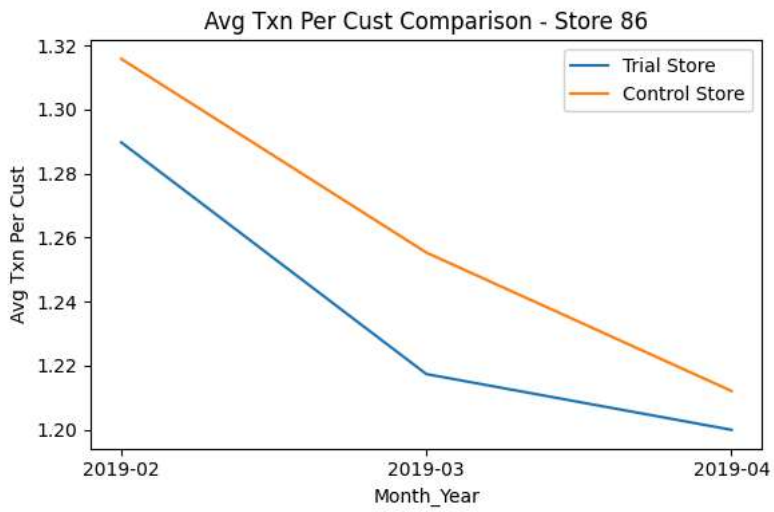



```
Trial Store :86,Control Store :155
```

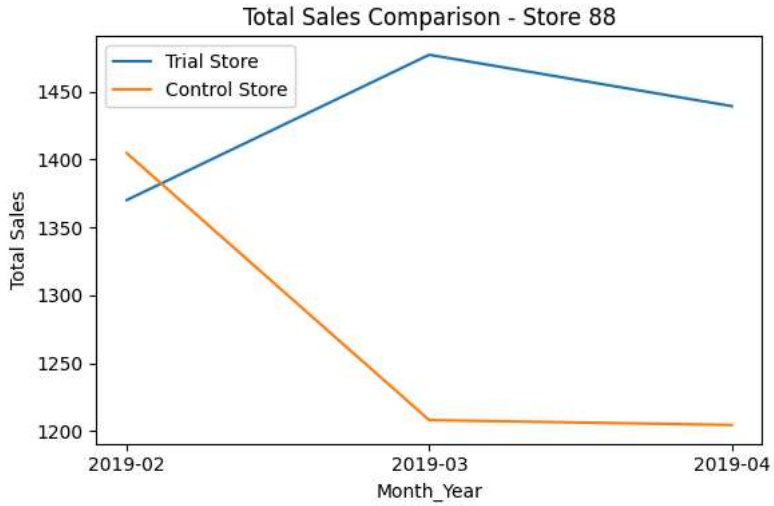


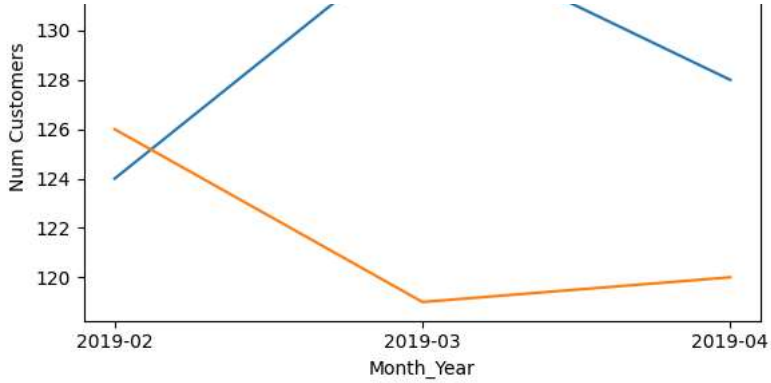


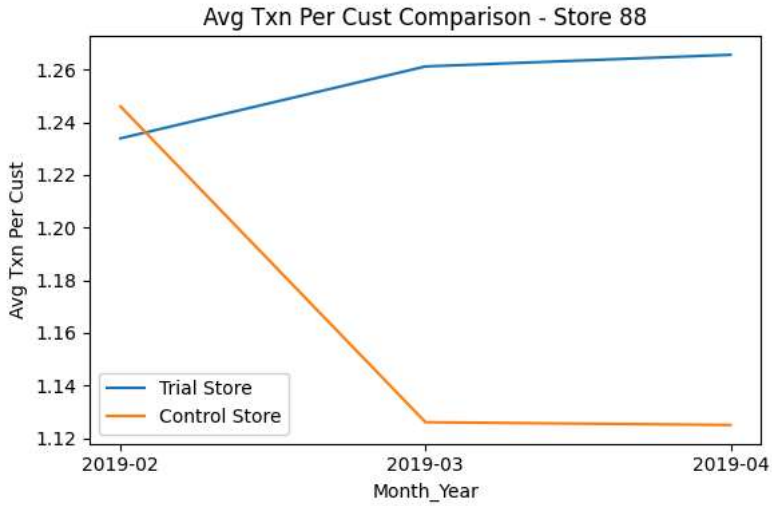





```
Trial Store :88,Control Store :237
```








```
from scipy.stats import ttest_ind
```

[ ] ↳ 2 cells hidden

hm

```
from scipy.stats import ttest_ind


for trial_store , control_store in control_stores.items():
    trial = monthly_metrics[(monthly_metrics['STORE_NBR']== trial_store)&
                            (monthly_metrics['MONTH_YEAR'].isin(trial_period))]['total_sales']

    control = monthly_metrics[(monthly_metrics['STORE_NBR']== control_store)&
                            (monthly_metrics['MONTH_YEAR'].isin(trial_period))]['total_sales']

    t_stat,p_value = ttest_ind(trial,control)
    print(f"Store {trial_store}:t-statistic = {t_stat:.3f},p-value = {p_value:.3f}")
```

```
Store 77:t-statistic = 2.104,p-value = 0.103
Store 86:t-statistic = 1.428,p-value = 0.227
Store 88:t-statistic = 2.137,p-value = 0.099
```

Start coding or generate with AI.

Start coding or generate with AI.