Quantium Analysis Task Two

# Importing Required Libraries.

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

sns.set_theme(style="darkgrid")

plt.rcParams['figure.figsize'] = [12, 6]
```

# Loading Data.

```
customers = pd.read_csv('/kaggle/input/quantium/QVI_purchase_behaviour.csv')
```

(purchase behviour data)

```
chips = pd.read_excel('/kaggle/input/quantium/QVI_transaction_data.xlsx')
```

(transactions data)

# Control Stores Metrics

```
grouped = merged_data.groupby([merged_data['STORE_NBR'],
merged_data['DATE'].dt.to_period('M')])


# Calculate metrics

measureOverTime = grouped.agg(

  total_sales=pd.NamedAgg(column='TOT_SALES', aggfunc='sum'),

  num_customers=pd.NamedAgg(column='LYLTY_CARD_NBR',
aggfunc=pd.Series.nunique),

  total_transactions=pd.NamedAgg(column='TXN_ID', aggfunc='count'),

  total_chips=pd.NamedAgg(column='PROD_QTY', aggfunc='sum'))
```

```
# Calculate additional metrics
measureOverTime['transactions_per_customer'] = measureOverTime['total_transactions'] / measureOverTime['num_customers']
measureOverTime['chips_per_customer'] = measureOverTime['total_chips'] / measureOverTime['num_customers']
measureOverTime['avg_price_per_unit'] = measureOverTime['total_sales'] / measureOverTime['total_chips']
measureOverTime = measureOverTime.reset_index()

# Extract year and month for filtering
measureOverTime['YEAR'] = measureOverTime['DATE'].dt.year
measureOverTime['MONTH'] = measureOverTime['DATE'].dt.month
measureOverTime['YEARMONTH'] = measureOverTime['YEAR'] * 100 + measureOverTime['MONTH']

# Identify stores with a full 12 months of observations
storesWithFullObs = measureOverTime.groupby('STORE_NBR').filter(lambda x: x['YEARMONTH'].nunique() == 12)['STORE_NBR'].unique()

# Filter to pre-trial period and stores with full observation periods
preTrialMeasures = measureOverTime[(measureOverTime['YEARMONTH'] < 201902) & (measureOverTime['STORE_NBR'].isin(storesWithFullObs))]
```

# Correlation Function

```
def calculate_correlations(measure_df,store, metric):
  """
  Parameters:
  measure_df (pd.DataFrame): The pre-trial measures dataframe.
  store (int): Trial Store Number.
  metric (str): The metric to compare.
```

```
    Returns:
    pd.DataFrame: Dataframe with store numbers and their respective correlations.
    """
    results = []
    store_numbers = measure_df['STORE_NBR'].unique()
    store_data = measure_df[measure_df['STORE_NBR'] == store]



    for control_store in store_numbers:
        if store != control_store:
            control_data = measure_df[measure_df['STORE_NBR'] == control_store]

            # Merge the data on YEARMONTH
            merged = store_data.merge(control_data, on='YEARMONTH', suffixes=('_store', '_control'))

            # Calculate the correlation
            correlation = merged[metric + '_store'].corr(merged[metric + '_control'])

            results.append({
                'store' : store,
                'control_store' : control_store,
                'correlation_'+metric : correlation
            })

    return pd.DataFrame(results)
```

# SMD Function

```
def calculate_smd(data, store_col, measure_col, treatment_store):
  """
  Calculate the standardized magnitude distance (SMD) between trial
  and control stores for a given measure.

  Parameters:
  data (pd.DataFrame): The dataframe containing the measures.
  store_col (str): The column name for store numbers.
  measure_col (str): The column name for the measure to compare.
  treatment_store (int or str): The treatment store number.
  control_store (int or str): The control store number.

  Returns:
  float: The standardized magnitude distance (SMD) between the treatment
      and control stores for the specified measure.
  """

  # Filter data for treatment and control stores
  results= []
  treatment_data = data[data[store_col] == treatment_store][measure_col]

  store_numbers = data['STORE_NBR'].unique()
  for control in store_numbers:
    if control != measure_col:
      control_data = data[data[store_col] == control][measure_col]
```

```
        # Calculate means and standard deviations
        mean_treatment = treatment_data.mean()
        mean_control = control_data.mean()
        sd_treatment = treatment_data.std()
        sd_control = control_data.std()

        # Calculate pooled standard deviation
        pooled_sd = ((sd_treatment**2 + sd_control**2) / 2)**0.5

        # Calculate SMD
        smd = (mean_treatment - mean_control) / pooled_sd
        smd = max(smd, 0)

        results.append({
            'store': treatment_store,
            'control_store': control,
            'smd': smd
          })
    results= pd.DataFrame(results)
    results['smd'] = results['smd']/results['smd'].max()
    return pd.DataFrame(results)
```

# Combined

```
store_77_corr_totsales = calculate_correlations(preTrialMeasures, 77, 'total_sales')
store_77_corr_ncustomers = calculate_correlations(preTrialMeasures, 77, 'num_customers')
```

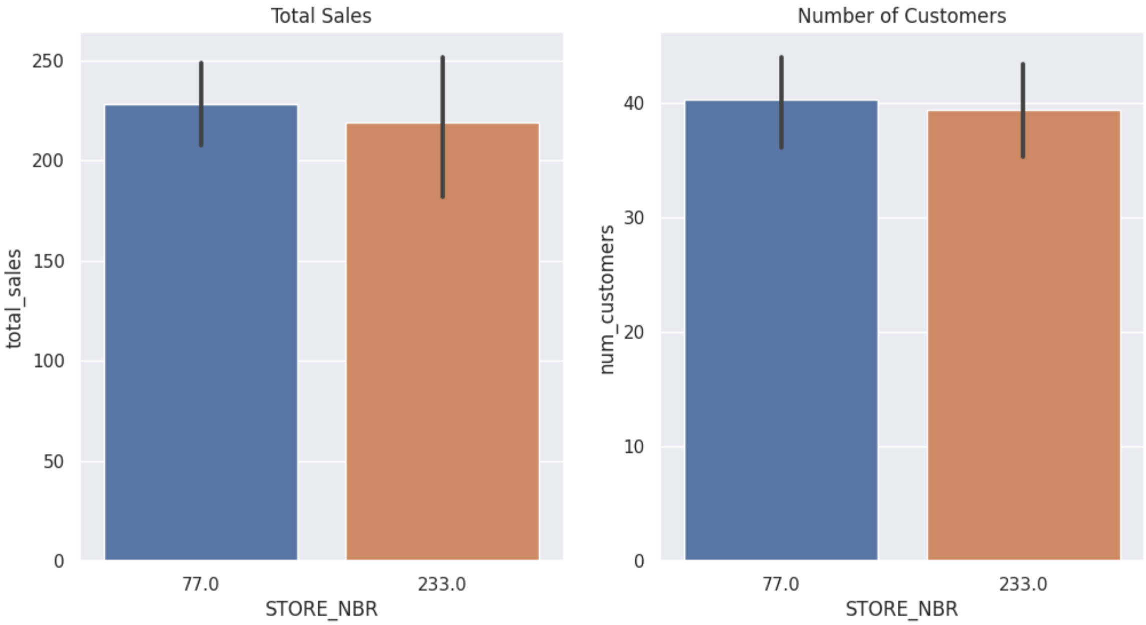
```
store_77_smd_value = calculate_smd(preTrialMeasures, 'STORE_NBR', 'total_sales', treatment_store=77)

store_77 = store_77_corr_totsales.merge(store_77_smd_value, on=['store','control_store'], how='inner')

store_77.sort_values(by=['correlation_num_customers','correlation_total_sales','smd'],ascending=False)
```

# Appropriate Control Store

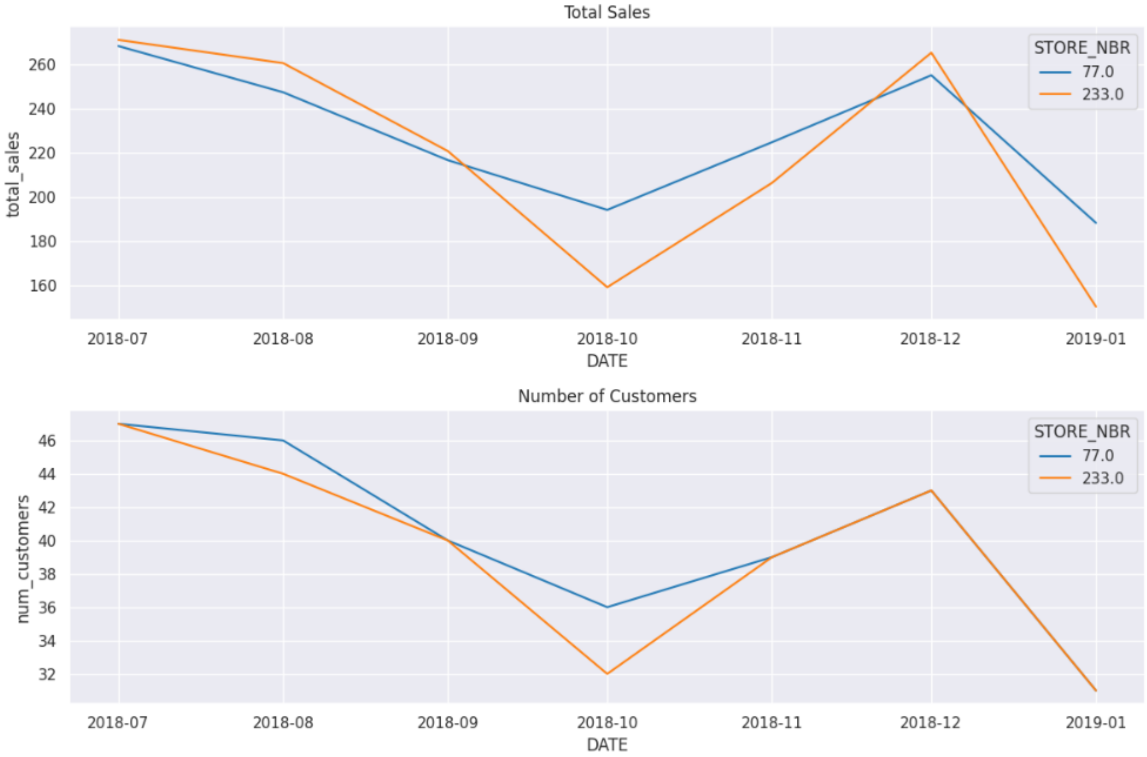
Control store : store number 233 with 0.965 , 0.973 correlation in number of customers and total sales

# Store 77 vs Store 233

```
store_77_233 = preTrialMeasures[preTrialMeasures['STORE_NBR'].isin([77,233])]

fig, axes = plt.subplots(1, 2)

sns.barplot(store_77_233,x='STORE_NBR',y='total_sales',ax=axes[0])

sns.barplot(store_77_233,x='STORE_NBR',y='num_customers',ax=axes[1])

axes[0].set_title('Total Sales')

axes[1].set_title('Number of Customers')
```



```
# Convert 'DATE' column from PeriodDtype to TimestampDtype

store_77_233['DATE'] = store_77_233['DATE'].dt.to_timestamp()


fig, axes = plt.subplots(2, 1, figsize=(12, 8))


sns.lineplot(data=store_77_233, x='DATE', y='total_sales', ax=axes[0],
hue='STORE_NBR',palette='tab10')

sns.lineplot(data=store_77_233, x='DATE', y='num_customers', ax=axes[1],
hue='STORE_NBR',palette='tab10')


axes[0].set_title('Total Sales')

axes[1].set_title('Number of Customers')
```



# Scaling Control Store Sales

```
trial_store = 77

control_store = 233
```

```
# Filtering the pre-trial period

control_store_sales = measureOverTime.query('STORE_NBR == 233').filter(['YEARMONTH', 'total_sales'], axis=1)

control_store_sales.rename(columns={'total_sales': 'store_233_sales'}, inplace=True)


trial_store_sales = measureOverTime.query('STORE_NBR == 77').filter(['YEARMONTH', 'total_sales'], axis=1)

trial_store_sales.rename(columns={'total_sales': 'store_77_sales'}, inplace=True)


sales_data = control_store_sales.merge(trial_store_sales,on='YEARMONTH',how='inner')

# Applying Scaling Factor

sales_data['store_233_sales'] = sales_data['store_233_sales']*scaling_factor_for_control_sales
```

# Percentage Difference

```
sales_data['pct_diff']=abs(((sales_data['store_233_sales']/sales_data['store_77_sales'])-1)*100)

sales_data['pct_diff'] = sales_data['pct_diff'].round(2)

pre_trial_sales_data= sales_data[sales_data['YEARMONTH'] <201902]

trial_sales_data= sales_data[sales_data['YEARMONTH'] >=201902]
```

Standard Deviation of Percentage Difference in Pre-Trial Period: 4.856300767338978,

While Standard Deviation of Percentage Difference in Trial Period: 12.562436865512995

# T-Test

```
from scipy.stats import t
```

```
# 1. Calculate the t-values for the trial months
trial_sales_data['t_value'] = (trial_sales_data['pct_diff'] - 0) /
pre_trial_sales_data['pct_diff'].std()

# 2. Determine the 95th percentile of the t-distribution
degreesOfFreedom = 7
critical_value = t.ppf(0.95, df=degreesOfFreedom)

# 3. Check for statistical significance
trial_sales_data['is_significant'] = np.abs(trial_sales_data['t_value']) > critical_value

print(trial_sales_data[['YEARMONTH', 't_value', 'is_significant']])
```

# Control Store 95, 5 percentiles

```
trial_store = 77
control_store = 233

pastSales = measureOverTime[(measureOverTime['STORE_NBR'] == trial_store) |
(measureOverTime['STORE_NBR'] == control_store)].copy()

# Assume stdDev is calculated based on some logic
stdDev = pastSales.groupby('STORE_NBR')['total_sales'].std().iloc[0]

# Control store 95th percentile
pastSales_Controls95 = pastSales[pastSales['STORE_NBR'] == control_store].copy()

pastSales_Controls95['total_sales'] = pastSales_Controls95['total_sales'] * (1 +
stdDev * 2)
pastSales_Controls95['STORE_NBR'] = 'Control 95th % confidence interval'
```

```
# Control store 5th percentile

pastSales_Controls5 = pastSales[pastSales['STORE_NBR'] == control_store].copy()

pastSales_Controls5['total_sales'] = pastSales_Controls5['total_sales'] * (1 - stdDev * 2)

pastSales_Controls5['STORE_NBR'] = 'Control 5th % confidence interval'


# Combine the data

trialAssessment = pd.concat([pastSales, pastSales_Controls95, pastSales_Controls5])


# Plot the data

# Prepare the subplots

sns.set(style="whitegrid")

fig, axes = plt.subplots(1, 2, figsize=(16, 8))


# Highlight the trial period

trial_period = (pd.to_datetime('201902', format='%Y%m'), pd.to_datetime('201905', format='%Y%m'))


# Plot the normal total sales on the first subplot

sns.lineplot(x='DATE', y='total_sales', hue='STORE_NBR', data=trialAssessment[trialAssessment['STORE_NBR'].isin(['77.0', '233.0'])], marker='o', ax=axes[0])

axes[0].axvspan(trial_period[0], trial_period[1], color='grey', alpha=0.2, label='Trial Period')

axes[0].set_title('Total Sales by Month (Normal Sales)')

axes[0].set_xlabel('Month of Operation')

axes[0].set_ylabel('Total Sales')
```

```
axes[0].legend(title='Store Type')

axes[0].tick_params(axis='x', rotation=45)


# Plot the 95th and 5th percentiles on the second subplot

sns.lineplot(x='DATE', y='total_sales', hue='STORE_NBR', data=trialAssessment[~trialAssessment['STORE_NBR'].isin(['77.0', '233.0'])], marker='o', ax=axes[1])

axes[1].axvspan(trial_period[0], trial_period[1], color='grey', alpha=0.2, label='Trial Period')

axes[1].set_title('Total Sales by Month (95th and 5th Percentiles)')

axes[1].set_xlabel('Month of Operation')

axes[1].set_ylabel('Total Sales')

axes[1].legend(title='Store Type')

axes[1].tick_params(axis='x', rotation=45)


plt.tight_layout()

plt.show()
```

# Num Of Customers

# Scaling

```
control_store_num_customers = measureOverTime.query('STORE_NBR == 233').filter(['YEARMONTH', 'num_customers'], axis=1)

control_store_num_customers.rename(columns={'num_customers': 'store_233_num_customers'}, inplace=True)


# Filtering the pre-trial period

pre_trial_trial_store_num_customers = preTrialMeasures[(preTrialMeasures['STORE_NBR'] == trial_store) &

                                (preTrialMeasures['YEARMONTH'] < 201902)]['num_customers'].sum()
```

```
pre_trial_control_store_num_customers = preTrialMeasures[(preTrialMeasures['STORE_NBR'] == control_store) &

                                    (preTrialMeasures['YEARMONTH'] < 201902)]['num_customers'].sum()


# Calculating the scaling factor

scaling_factor_for_num_customers = pre_trial_trial_store_num_customers / pre_trial_control_store_num_customers

scaling_factor_for_num_customers


trial_store_num_customers = measureOverTime.query('STORE_NBR == 77').filter(['YEARMONTH', 'num_customers'], axis=1)

trial_store_num_customers.rename(columns={'num_customers': 'store_77_num_customers'}, inplace=True)


num_customers_data = control_store_num_customers.merge(trial_store_num_customers,on='YEARMONTH',how='inner')

# Applying Scaling Factornum_customers

num_customers_data['store_233_num_customers'] =num_customers_data['store_233_num_customers']*scaling_factor_for_num_customers

num_customers_data['pct_diff']=abs(((num_customers_data['store_233_num_customers']/num_customers_data['store_77_num_customers'])-1)*100)

num_customers_data['pct_diff'] = num_customers_data['pct_diff'].round(2)

num_customers_data
```

# Percentiles

```
# Calculate standard deviation based on pre-trial percentage difference

std_dev = np.std(num_customers_data[num_customers_data['YEARMONTH'] < 201902]['pct_diff'])
```

```
# Define degrees of freedom (length of pre-trial period - 1)
degrees_of_freedom =
len(num_customers_data[num_customers_data['YEARMONTH'] < 201902]) - 1


# Define trial period
trial_period_start = pd.to_datetime('2019-02-01')
trial_period_end = pd.to_datetime('2019-05-01')
# Compute 95th and 5th percentiles for control store
control_95th = num_customers_data.copy()
control_95th['store_233_num_customers'] =
control_95th['store_233_num_customers'] * (1 + std_dev * 2)
control_95th['STORE_NBR'] = 'Control 95th % confidence interval'


control_5th = num_customers_data.copy()
control_5th['store_233_num_customers'] =
control_5th['store_233_num_customers'] * (1 - std_dev * 2)
control_5th['STORE_NBR'] = 'Control 5th % confidence interval'


# Combine all data
trial_assessment = pd.concat([num_customers_data, control_95th, control_5th])
```

# Plot

```
# Prepare subplots
fig, axes = plt.subplots(1, 2, figsize=(16, 8))


# Plot the normal number of customers on the first subplot
axes[0].plot(num_customers_data['YEARMONTH'],
num_customers_data['store_77_num_customers'], marker='o', label='Trial Store')
```

```
axes[0].plot(num_customers_data['YEARMONTH'], num_customers_data['store_233_num_customers'], marker='o', label='Scaled Control Store')


axes[0].axvspan(201902, 201905, color='grey', alpha=0.2, label='Trial Period')

axes[0].set_title('Number of Customers by Month (Normal)')

axes[0].set_xlabel('YEARMONTH')

axes[0].set_ylabel('Number of Customers')

axes[0].legend(title='Store Type')

axes[0].tick_params(axis='x', rotation=45)


# Plot the 95th and 5th percentiles on the second subplot

axes[1].plot(control_95th['YEARMONTH'], control_95th['store_233_num_customers'], marker='o', label='Control 95th % confidence interval')

axes[1].plot(control_5th['YEARMONTH'], control_5th['store_233_num_customers'], marker='o', label='Control 5th % confidence interval')


axes[1].axvspan(201902, 201905, color='grey', alpha=0.2, label='Trial Period')

axes[1].set_title('Number of Customers by Month (95th and 5th Percentiles)')

axes[1].set_xlabel('YEARMONTH')

axes[1].set_ylabel('Number of Customers')

axes[1].legend(title='Store Type')

axes[1].tick_params(axis='x', rotation=45)
```

# Trial Store 86

```
store_86_corr_totsales = calculate_correlations(preTrialMeasures, 86, 'total_sales')

store_86_corr_ncustomers = calculate_correlations(preTrialMeasures, 86, 'num_customers')
```

```
store_86_smd_value = calculate_smd(preTrialMeasures, 'STORE_NBR', 'total_sales', treatment_store=86)
store_86 = store_86_corr_totsales.merge(store_86_smd_value, on=['store','control_store'], how='inner')
store_86 = store_86_corr_ncustomers.merge(store_86, on=['store','control_store'], how='inner')

store_86.sort_values(by=['correlation_num_customers','correlation_total_sales'],ascending=False,inplace=True)
store_86
```

```
store_86_155 = preTrialMeasures[preTrialMeasures['STORE_NBR'].isin([86, 155])]

# Convert 'DATE' column from PeriodDtype to TimestampDtype
store_86_155['DATE'] = store_86_155['DATE'].dt.to_timestamp()

fig, axes = plt.subplots(2, 1, figsize=(16, 8))

sns.lineplot(data=store_86_155, x='DATE', y='total_sales', ax=axes[0], hue='STORE_NBR',palette='tab10')
sns.lineplot(data=store_86_155, x='DATE', y='num_customers', ax=axes[1], hue='STORE_NBR',palette='tab10')

axes[0].set_title('Total Sales')
axes[1].set_title('Number of Customers')

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

```
fig, axes = plt.subplots(1, 2, figsize=(12, 8))


sns.barplot(store_86_155,x='STORE_NBR',y='total_sales',ax=axes[0])

sns.barplot(store_86_155,x='STORE_NBR',y='num_customers',ax=axes[1])

axes[0].set_title('Total Sales')

axes[1].set_title('Number of Customers')
```

```
trial_store = 86

control_store = 155


# Filtering the pre-trial period

pre_trial_trial_store_sales = preTrialMeasures[(preTrialMeasures['STORE_NBR'] ==
trial_store) &

                              (preTrialMeasures['YEARMONTH'] <
201902)]['total_sales'].sum()

pre_trial_control_store_sales = preTrialMeasures[(preTrialMeasures['STORE_NBR']
== control_store) &

                                (preTrialMeasures['YEARMONTH'] <
201902)]['total_sales'].sum()


# Calculating the scaling factor

scaling_factor_for_control_sales = pre_trial_trial_store_sales /
pre_trial_control_store_sales

scaling_factor_for_control_sales
```

```
control_store_sales = measureOverTime.query('STORE_NBR == 155).filter(['YEARMONTH', 'total_sales'], axis=1)

control_store_sales.rename(columns={'total_sales': 'store_155_sales'}, inplace=True)


trial_store_sales = measureOverTime.query('STORE_NBR == 86).filter(['YEARMONTH', 'total_sales'], axis=1)

trial_store_sales.rename(columns={'total_sales': 'store_86_sales'}, inplace=True)


sales_data = control_store_sales.merge(trial_store_sales,on='YEARMONTH',how='inner')

# Applying Scaling Factor

sales_data['store_155_sales'] = sales_data['store_155_sales']*scaling_factor_for_control_sales


sales_data['pct_diff']=abs(((sales_data['store_155_sales']/sales_data['store_86_sales'])-1)*100)

sales_data['pct_diff'] = sales_data['pct_diff'].round(2)

sales_data


pre_trial_sales_data= sales_data[sales_data['YEARMONTH'] <201902]

trial_sales_data= sales_data[sales_data['YEARMONTH'] >=201902]

print(f'''Standard Deviation of Percentage Difference in Pre-Trial Period: {pre_trial_sales_data['pct_diff'].std()},

While Standard Deviation of Percentage Difference in Trial Period: {trial_sales_data['pct_diff'].std()}''')

from scipy.stats import t


# 1. Calculate the t-values for the trial months
```

```
trial_sales_data['t_value'] = (trial_sales_data['pct_diff'] - 0) /
pre_trial_sales_data['pct_diff'].std()

# 2. Determine the 95th percentile of the t-distribution
degreesOfFreedom = 7
critical_value = t.ppf(0.95, df=degreesOfFreedom)

# 3. Check for statistical significance
trial_sales_data['is_significant'] = np.abs(trial_sales_data['t_value']) > critical_value

print(trial_sales_data[['YEARMONTH', 't_value', 'is_significant']])
```

```
# Create new variables in the DataFrame
trial_store = 86
control_store = 155

pastSales = measureOverTime[(measureOverTime['STORE_NBR'] == trial_store) |
(measureOverTime['STORE_NBR'] == control_store)].copy()

# Assume stdDev is calculated based on some logic
stdDev = pastSales.groupby('STORE_NBR')['total_sales'].std().iloc[0]

# Control store 95th percentile
pastSales_Controls95 = pastSales[pastSales['STORE_NBR'] == control_store].copy()

pastSales_Controls95['total_sales'] = pastSales_Controls95['total_sales'] * (1 +
stdDev * 2)
```

```
pastSales_Controls95['STORE_NBR'] = 155.95

# Control store 5th percentile
pastSales_Controls5 = pastSales[pastSales['STORE_NBR'] == control_store].copy()
pastSales_Controls5['total_sales'] = pastSales_Controls5['total_sales'] * (1 - stdDev * 2)
pastSales_Controls5['STORE_NBR'] = 155.05

# Combine the data
trialAssessment = pd.concat([pastSales, pastSales_Controls95, pastSales_Controls5])
trialAssessment.head()
```

```
normal_sales_df = trialAssessment[trialAssessment['STORE_NBR'].isin([77.0, 233.0])]
percentile_sales_df= trialAssessment[~trialAssessment['STORE_NBR'].isin([77.0, 233.0])]

normal_sales_df['DATE'] = normal_sales_df['DATE'].dt.to_timestamp()
percentile_sales_df['DATE'] = percentile_sales_df['DATE'].dt.to_timestamp()
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Prepare the subplots
sns.set(style="whitegrid")
fig, axes = plt.subplots(2, 1, figsize=(16, 8))

# Highlight the trial period
trial_period = (pd.to_datetime('201902', format='%Y%m'), pd.to_datetime('201905', format='%Y%m'))
```

```
# Plot the normal total sales on the first subplot
sns.lineplot(x='DATE', y='total_sales', hue='STORE_NBR', data=normal_sales_df, marker='o', ax=axes[0])
axes[0].axvspan(trial_period[0], trial_period[1], color='grey', alpha=0.2, label='Trial Period')
axes[0].set_title('Total Sales by Month (Normal Sales)')
axes[0].set_xlabel('Month of Operation')
axes[0].set_ylabel('Total Sales')
axes[0].legend(title='Store Type')
axes[0].tick_params(axis='x', rotation=45)

# Plot the 95th and 5th percentiles on the second subplot
sns.lineplot(x='DATE', y='total_sales', hue='STORE_NBR', data=percentile_sales_df, marker='o', ax=axes[1])
axes[1].axvspan(trial_period[0], trial_period[1], color='grey', alpha=0.2, label='Trial Period')
axes[1].set_title('Total Sales by Month (95th and 5th Percentiles)')
axes[1].set_xlabel('Month of Operation')
axes[1].set_ylabel('Total Sales')
axes[1].legend(title='Store Type')
axes[1].tick_params(axis='x', rotation=45)

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

```
control_store_num_customers = measureOverTime.query('STORE_NBR == 155).filter(['YEARMONTH', 'num_customers'], axis=1)
```

```
control_store_num_customers.rename(columns={'num_customers': 'store_155_num_customers'}, inplace=True)


# Filtering the pre-trial period

pre_trial_trial_store_num_customers = preTrialMeasures[(preTrialMeasures['STORE_NBR'] == trial_store) &

                                  (preTrialMeasures['YEARMONTH'] < 201902)]['num_customers'].sum()

pre_trial_control_store_num_customers = preTrialMeasures[(preTrialMeasures['STORE_NBR'] == control_store) &

                                   (preTrialMeasures['YEARMONTH'] < 201902)]['num_customers'].sum()


# Calculating the scaling factor

scaling_factor_for_num_customers = pre_trial_trial_store_num_customers / pre_trial_control_store_num_customers

scaling_factor_for_num_customers


trial_store_num_customers = measureOverTime.query('STORE_NBR == 86).filter(['YEARMONTH', 'num_customers'], axis=1)

trial_store_num_customers.rename(columns={'num_customers': 'store_86_num_customers'}, inplace=True)


num_customers_data = control_store_num_customers.merge(trial_store_num_customers,on='YEARMONTH',how='inner')

# Applying Scaling Factornum_customers

num_customers_data['store_155_num_customers'] =num_customers_data['store_155_num_customers']*scaling_factor_for_num_customers

num_customers_data['pct_diff']=abs(((num_customers_data['store_155_num_customers']/num_customers_data['store_86_num_customers'])-1)*100)
```

```
num_customers_data['pct_diff'] = num_customers_data['pct_diff'].round(2)

num_customers_data
```

```
# Calculate standard deviation based on pre-trial percentage difference

std_dev = np.std(num_customers_data[num_customers_data['YEARMONTH'] < 201902]['pct_diff'])


# Define degrees of freedom (length of pre-trial period - 1)

degrees_of_freedom = len(num_customers_data[num_customers_data['YEARMONTH'] < 201902]) - 1


# Define trial period

trial_period_start = pd.to_datetime('2019-02-01')

trial_period_end = pd.to_datetime('2019-05-01')
```

```
# Compute 95th and 5th percentiles for control store

control_95th = num_customers_data.copy()

control_95th['store_155_num_customers'] = control_95th['store_155_num_customers'] * (1 + std_dev * 2)

control_95th['STORE_NBR'] = 'Control 95th % confidence interval'


control_5th = num_customers_data.copy()

control_5th['store_155_num_customers'] = control_5th['store_155_num_customers'] * (1 - std_dev * 2)

control_5th['STORE_NBR'] = 'Control 5th % confidence interval'


# Combine all data

trial_assessment = pd.concat([num_customers_data, control_95th, control_5th])
```

```
import matplotlib.pyplot as plt
```

```
# Prepare subplots
fig, axes = plt.subplots(2, 1, figsize=(16, 8))

# Plot the normal number of customers on the first subplot
axes[0].plot(num_customers_data['YEARMONTH'], num_customers_data['store_86_num_customers'], marker='o', label='Trial Store')
axes[0].plot(num_customers_data['YEARMONTH'], num_customers_data['store_155_num_customers'], marker='o', label='Scaled Control Store')

axes[0].axvspan(201902, 201905, color='grey', alpha=0.2, label='Trial Period')
axes[0].set_title('Number of Customers by Month (Normal)')
axes[0].set_xlabel('YEARMONTH')
axes[0].set_ylabel('Number of Customers')
axes[0].legend(title='Store Type')
axes[0].tick_params(axis='x', rotation=45)

# Plot the 95th and 5th percentiles on the second subplot
axes[1].plot(control_95th['YEARMONTH'], control_95th['store_155_num_customers'], marker='o', label='Control 95th % confidence interval')
axes[1].plot(control_5th['YEARMONTH'], control_5th['store_155_num_customers'], marker='o', label='Control 5th % confidence interval')

axes[1].axvspan(201902, 201905, color='grey', alpha=0.2, label='Trial Period')
axes[1].set_title('Number of Customers by Month (95th and 5th Percentiles)')
axes[1].set_xlabel('YEARMONTH')
axes[1].set_ylabel('Number of Customers')
axes[1].legend(title='Store Type')
```

```
axes[1].tick_params(axis='x', rotation=45)


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

# Store 88

```
store_88_corr_totsales = calculate_correlations(preTrialMeasures, 88, 'total_sales')
store_88_corr_ncustomers = calculate_correlations(preTrialMeasures, 88, 'num_customers')


store_88_smd_value = calculate_smd(preTrialMeasures, 'STORE_NBR', 'total_sales', treatment_store=88)
store_88 = store_88_corr_totsales.merge(store_88_smd_value, on=['store','control_store'], how='inner')
store_88 = store_88_corr_ncustomers.merge(store_88, on=['store','control_store'], how='inner')


store_88.sort_values(by=['correlation_num_customers','correlation_total_sales'],ascending=False,inplace=True)
store_88
```

```
store_88_237 = preTrialMeasures[preTrialMeasures['STORE_NBR'].isin([88, 237])]


# Convert 'DATE' column from PeriodDtype to TimestampDtype
store_88_237['DATE'] = store_88_237['DATE'].dt.to_timestamp()
```

```
fig, axes = plt.subplots(2, 1, figsize=(16, 8))

sns.lineplot(data=store_88_237, x='DATE', y='total_sales', ax=axes[0],
hue='STORE_NBR',palette='tab10')
sns.lineplot(data=store_88_237, x='DATE', y='num_customers', ax=axes[1],
hue='STORE_NBR',palette='tab10')

axes[0].set_title('Total Sales')
axes[1].set_title('Number of Customers')

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

```
fig, axes = plt.subplots(1, 2, figsize=(12, 8))

sns.barplot(store_88_237,x='STORE_NBR',y='total_sales',ax=axes[0])
sns.barplot(store_88_237,x='STORE_NBR',y='num_customers',ax=axes[1])
axes[0].set_title('Total Sales')
axes[1].set_title('Number of Customers')
```

```
trial_store = 88
control_store = 237
```

```
# Filtering the pre-trial period

pre_trial_trial_store_sales =
preTrialMeasures[(preTrialMeasures['STORE_NBR'] == trial_store) &

                              (preTrialMeasures['YEARMONTH'] <
201902)]['total_sales'].sum()

pre_trial_control_store_sales =
preTrialMeasures[(preTrialMeasures['STORE_NBR'] == control_store) &

                               (preTrialMeasures['YEARMONTH'] <
201902)]['total_sales'].sum()


# Calculating the scaling factor

scaling_factor_for_control_sales = pre_trial_trial_store_sales /
pre_trial_control_store_sales

scaling_factor_for_control_sales
```

```
control_store_sales = measureOverTime.query('STORE_NBR ==
237).filter(['YEARMONTH', 'total_sales'], axis=1)

control_store_sales.rename(columns={'total_sales': 'store_237_sales'},
inplace=True)


trial_store_sales = measureOverTime.query('STORE_NBR ==
88).filter(['YEARMONTH', 'total_sales'], axis=1)

trial_store_sales.rename(columns={'total_sales': 'store_88_sales'},
inplace=True)


sales_data =
control_store_sales.merge(trial_store_sales,on='YEARMONTH',how='inner')

# Applying Scaling Factor
```

```
sales_data['store_237_sales'] =
sales_data['store_237_sales']*scaling_factor_for_control_sales
```

```
sales_data['pct_diff']=abs(((sales_data['store_237_sales']/sales_data['store_
88_sales'])-1)*100)

sales_data['pct_diff'] = sales_data['pct_diff'].round(2)

sales_data
```

```
pre_trial_sales_data= sales_data[sales_data['YEARMONTH'] <201902]

trial_sales_data= sales_data[sales_data['YEARMONTH'] >=201902]

print(f'''Standard Deviation of Percentage Difference in Pre-Trial Period:
{pre_trial_sales_data['pct_diff'].std()},

While Standard Deviation of Percentage Difference in Trial Period:
{trial_sales_data['pct_diff'].std()}''')
```

```
from scipy.stats import t


# 1. Calculate the t-values for the trial months

trial_sales_data['t_value'] = (trial_sales_data['pct_diff'] - 0) /
pre_trial_sales_data['pct_diff'].std()


# 2. Determine the 95th percentile of the t-distribution

degreesOfFreedom = 7

critical_value = t.ppf(0.95, df=degreesOfFreedom)


# 3. Check for statistical significance

trial_sales_data['is_significant'] = np.abs(trial_sales_data['t_value']) >
critical_value
```

```
print(trial_sales_data[['YEARMONTH', 't_value', 'is_significant']])
```

```
# Create new variables in the DataFrame
trial_store = 88
control_store = 237

pastSales = measureOverTime[(measureOverTime['STORE_NBR'] == trial_store) | (measureOverTime['STORE_NBR'] == control_store)].copy()

# Assume stdDev is calculated based on some logic
stdDev = pastSales.groupby('STORE_NBR')['total_sales'].std().iloc[0]

# Control store 95th percentile
pastSales_Controls95 = pastSales[pastSales['STORE_NBR'] == control_store].copy()

pastSales_Controls95['total_sales'] = pastSales_Controls95['total_sales'] * (1 + stdDev * 2)
pastSales_Controls95['STORE_NBR'] = 237.95

# Control store 5th percentile
pastSales_Controls5 = pastSales[pastSales['STORE_NBR'] == control_store].copy()
pastSales_Controls5['total_sales'] = pastSales_Controls5['total_sales'] * (1 - stdDev * 2)
pastSales_Controls5['STORE_NBR'] = 237.05
```

```
# Combine the data

trialAssessment = pd.concat([pastSales, pastSales_Controls95,
pastSales_Controls5])

trialAssessment.head()
```

```
normal_sales_df =
trialAssessment[trialAssessment['STORE_NBR'].isin([77.0, 233.0])]

percentile_sales_df=
trialAssessment[~trialAssessment['STORE_NBR'].isin([77.0, 233.0])]


normal_sales_df['DATE'] = normal_sales_df['DATE'].dt.to_timestamp()

percentile_sales_df['DATE'] = percentile_sales_df['DATE'].dt.to_timestamp()
```

```
import matplotlib.pyplot as plt

import seaborn as sns


# Prepare the subplots

sns.set(style="whitegrid")

fig, axes = plt.subplots(2, 1, figsize=(16, 8))


# Highlight the trial period

trial_period = (pd.to_datetime('201902', format='%Y%m'),
pd.to_datetime('201905', format='%Y%m'))


# Plot the normal total sales on the first subplot

sns.lineplot(x='DATE', y='total_sales', hue='STORE_NBR',
data=normal_sales_df, marker='o', ax=axes[0])

axes[0].axvspan(trial_period[0], trial_period[1], color='grey', alpha=0.2,
label='Trial Period')
```

```
axes[0].set_title('Total Sales by Month (Normal Sales)')

axes[0].set_xlabel('Month of Operation')

axes[0].set_ylabel('Total Sales')

axes[0].legend(title='Store Type')

axes[0].tick_params(axis='x', rotation=45)


# Plot the 95th and 5th percentiles on the second subplot

sns.lineplot(x='DATE', y='total_sales', hue='STORE_NBR', data=percentile_sales_df, marker='o', ax=axes[1])

axes[1].axvspan(trial_period[0], trial_period[1], color='grey', alpha=0.2, label='Trial Period')

axes[1].set_title('Total Sales by Month (95th and 5th Percentiles)')

axes[1].set_xlabel('Month of Operation')

axes[1].set_ylabel('Total Sales')

axes[1].legend(title='Store Type')

axes[1].tick_params(axis='x', rotation=45)


plt.tight_layout()

plt.show()
```

```
control_store_num_customers = measureOverTime.query('STORE_NBR == 237).filter(['YEARMONTH', 'num_customers'], axis=1)

control_store_num_customers.rename(columns={'num_customers': 'store_237_num_customers'}, inplace=True)


# Filtering the pre-trial period
```

```
pre_trial_trial_store_num_customers = preTrialMeasures[(preTrialMeasures['STORE_NBR'] == trial_store) &
                                  (preTrialMeasures['YEARMONTH'] < 201902)]['num_customers'].sum()
pre_trial_control_store_num_customers = preTrialMeasures[(preTrialMeasures['STORE_NBR'] == control_store) &
                                  (preTrialMeasures['YEARMONTH'] < 201902)]['num_customers'].sum()

# Calculating the scaling factor
scaling_factor_for_num_customers = pre_trial_trial_store_num_customers / pre_trial_control_store_num_customers
scaling_factor_for_num_customers

trial_store_num_customers = measureOverTime.query('STORE_NBR == 88).filter(['YEARMONTH', 'num_customers'], axis=1)
trial_store_num_customers.rename(columns={'num_customers': 'store_88_num_customers'}, inplace=True)

num_customers_data = control_store_num_customers.merge(trial_store_num_customers,on='YEARMONTH',how='inner')
# Applying Scaling Factornum_customers
num_customers_data['store_237_num_customers'] =num_customers_data['store_237_num_customers']*scaling_factor_for_num_customers
num_customers_data['pct_diff']=abs(((num_customers_data['store_237_num_customers']/num_customers_data['store_88_num_customers'])-1)*100)
num_customers_data['pct_diff'] = num_customers_data['pct_diff'].round(2)
num_customers_data
```

```
# Calculate standard deviation based on pre-trial percentage difference

std_dev =
np.std(num_customers_data[num_customers_data['YEARMONTH'] <
201902]['pct_diff'])


# Define degrees of freedom (length of pre-trial period - 1)

degrees_of_freedom =
len(num_customers_data[num_customers_data['YEARMONTH'] < 201902]) -
1


# Define trial period

trial_period_start = pd.to_datetime('2019-02-01')

trial_period_end = pd.to_datetime('2019-05-01')
```

```
# Compute 95th and 5th percentiles for control store

control_95th = num_customers_data.copy()

control_95th['store_237_num_customers'] =
control_95th['store_237_num_customers'] * (1 + std_dev * 2)

control_95th['STORE_NBR'] = 'Control 95th % confidence interval'


control_5th = num_customers_data.copy()

control_5th['store_237_num_customers'] =
control_5th['store_237_num_customers'] * (1 - std_dev * 2)

control_5th['STORE_NBR'] = 'Control 5th % confidence interval'


# Combine all data

trial_assessment = pd.concat([num_customers_data, control_95th,
control_5th])
```

```
import matplotlib.pyplot as plt


# Prepare subplots
fig, axes = plt.subplots(2, 1, figsize=(16, 8))


# Plot the normal number of customers on the first subplot
axes[0].plot(num_customers_data['YEARMONTH'], num_customers_data['store_88_num_customers'], marker='o', label='Trial Store')
axes[0].plot(num_customers_data['YEARMONTH'], num_customers_data['store_237_num_customers'], marker='o', label='Scaled Control Store')


axes[0].axvspan(201902, 201905, color='grey', alpha=0.2, label='Trial Period')
axes[0].set_title('Number of Customers by Month (Normal)')
axes[0].set_xlabel('YEARMONTH')
axes[0].set_ylabel('Number of Customers')
axes[0].legend(title='Store Type')
axes[0].tick_params(axis='x', rotation=45)


# Plot the 95th and 5th percentiles on the second subplot
axes[1].plot(control_95th['YEARMONTH'], control_95th['store_237_num_customers'], marker='o', label='Control 95th % confidence interval')
axes[1].plot(control_5th['YEARMONTH'], control_5th['store_237_num_customers'], marker='o', label='Control 5th % confidence interval')
```

```
axes[1].axvspan(201902, 201905, color='grey', alpha=0.2, label='Trial Period')

axes[1].set_title('Number of Customers by Month (95th and 5th Percentiles)')

axes[1].set_xlabel('YEARMONTH')

axes[1].set_ylabel('Number of Customers')

axes[1].legend(title='Store Type')

axes[1].tick_params(axis='x', rotation=45)


plt.tight_layout()

plt.show()
```