Task 2

Problem Context

Continuing on the task, a trial was conducted in stores: 77, 86, 88.

To determine whether they performed well compared to control stores.

This findings are needed by management team to present to client: the chips manufacturer.

See task 1 for more additional details.

In [756…

```
# Approach.
# 1. To try and understand the case presented.
# 2. To do the analysis as seen best fit, based on know how; intuitive un
    # Own interpertation.
    # Extend with solution template/new ideas from it/not considered.
# 3. Upload Solution.
# 4. See solution, provided.

# Please note a case study is open to interpretation.
# Also, whilst this task does involve some experimentation and testing.
# The notion is to compare and analyse based on metrics.
```

In [757…

```
# -- outline  --
# EDA
# Picking Control Stores.
# Comparing Control Stores to trial stores performance.  (vis, analysis.)
# Evaluating Key findings and conclusion.
# Improvements, limitations, etc..
```

1. Setup

In [758...

```
#Importing modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp

import os
import re
from collections import namedtuple
from IPython.display import display

# Setting Options
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',10)

def toggle_option(n:str=['all','10'],set_option_str:str=["display.max_col
    print('Set: ', set_option_str, ' to ', n)
    n=None if n == 'all' else int(n)
    pd.set_option(set_option_str,n)
    print('\n')
```

In [762...

```
# Loading in dataset, processed at the end of task 1.
data_set_df = pd.read_csv(f"{os.path.join(os.getcwd(),'quantium_dataset_p
```

In [763...

```
data_set_df = data_set_df.drop('Unnamed: 0',axis=1)
#data_set_df = data_set_df.iloc[:,1:]
```

In [764...

```
display(data_set_df.head())
stores = [77,86,88]
```

	Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationship_Ty
0	1000	Young Singles/Couples	Premium	Young	Singles/Coupl
1	1002	Young Singles/Couples	Mainstream	Young	Singles/Coupl
2	1003	Young Families	Budget	Young	Famili
3	1003	Young Families	Budget	Young	Famili
4	1004	Older Singles/Couples	Mainstream	Older	Singles/Coupl

2. EDA

In [765…

```
data_set_df.info()
toggle_option('all','display.max_rows')

# Numeric Columns Summary.
display((
    data_set_df
        .query(f'Store_Number.isin({stores})')
        .groupby('Store_Number')
        .describe()
        .T
))

# Object Columns list.
cols_objects = (
    ['Store_Number'] +

    list(
        data_set_df
            .select_dtypes('object')
            .columns
        )
)

# Object Columns Summary.
display((
    data_set_df[cols_objects]
        .query(f'Store_Number.isin({stores})')
        .groupby('Store_Number')
        .describe()
))


toggle_option(10,'display.max_rows')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 213986 entries, 0 to 213985
Data columns (total 19 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   Loyalty_Card_Number         213986 non-null  int64  
 1   Life_Stage                  213986 non-null  object 
 2   Card_Subscription           213986 non-null  object 
 3   Age_Group                   213986 non-null  object 
 4   Relationship_Type           213986 non-null  object 
 5   Date                        213986 non-null  object 
 6   Store_Number                213986 non-null  int64  
 7   Taxation_Id                 213986 non-null  int64  
 8   Product_Number              213986 non-null  int64  
 9   Product_Name                213986 non-null  object 
 10  Product_Quantity            213986 non-null  int64  
 11  Total_Sales                 213986 non-null  float64
 12  Product_Price               213986 non-null  float64
 13  Product_Weight_Grams        213986 non-null  int64  
 14  Brand_Name                  213986 non-null  object 
 15  Product_Price_Per_100_Grams  213986 non-null  float64
 16  Year                        213986 non-null  int64  
 17  Month                       213986 non-null  int64  
 18  Tax_Paid                    213986 non-null  float64
dtypes: float64(4), int64(8), object(7)
memory usage: 31.0+ MB
Set:  display.max_rows  to  all
```

	Store_Number	77	86	[illegible]
Loyalty_Card_Number	**count**	4.710000e+02	1259.000000	1.497000e+
	mean	1.585728e+05	87551.044480	1.035363e+
	std	4.206750e+05	9833.310412	1.860348e+
	min	7.700000e+04	86000.000000	8.800000e+
	25%	7.713200e+04	86062.500000	8.809300e+
	50%	7.726800e+04	86126.000000	8.818800e+
	75%	7.739950e+04	86190.000000	8.828300e+
	max	2.330501e+06	155510.000000	2.373711e+
Taxation_Id	**count**	4.710000e+02	1259.000000	1.497000e+
	mean	8.101963e+04	86357.654488	8.973612e+
	std	3.016587e+04	10014.633283	6.144563e+
	min	7.491000e+04	84137.000000	8.622000e+
	25%	7.505650e+04	84528.500000	8.668100e+
	50%	7.520300e+04	84922.000000	8.715800e+
	75%	7.534050e+04	85317.000000	8.763800e+
	max	2.367800e+05	155718.000000	2.415841e+
Product_Number	**count**	4.710000e+02	1259.000000	1.497000e+
	mean	5.596603e+01	56.523431	5.228724e+

	std	3.313170e+01	33.844755	3.297289e+
	min	1.000000e+00	1.000000	2.000000e+
	25%	2.700000e+01	26.000000	2.600000e+
	50%	5.800000e+01	58.000000	4.700000e+
	75%	8.300000e+01	85.000000	7.800000e+
	max	1.140000e+02	114.000000	1.140000e+
Product_Quantity	count	4.710000e+02	1259.000000	1.497000e+
	mean	1.547771e+00	1.988880	1.985304e+
	std	5.150401e-01	0.203007	2.095269e-
	min	1.000000e+00	1.000000	1.000000e+
	25%	1.000000e+00	2.000000	2.000000e+
	50%	2.000000e+00	2.000000	2.000000e+
	75%	2.000000e+00	2.000000	2.000000e+
	max	4.000000e+00	5.000000	5.000000e+
Total_Sales	count	4.710000e+02	1259.000000	1.497000e+
	mean	5.406157e+00	6.929349	8.680995e+
	std	2.431650e+00	2.261652	1.827326e+
	min	1.700000e+00	1.900000	3.250000e+
	25%	3.400000e+00	5.400000	7.400000e+
	50%	5.200000e+00	6.200000	8.400000e+
	75%	6.600000e+00	8.800000	9.200000e+
	max	1.520000e+01	16.800000	2.280000e+
Product_Price	count	4.710000e+02	1259.000000	1.497000e+
	mean	3.532909e+00	3.487808	4.371677e+
	std	1.129526e+00	1.100732	7.937734e-
	min	1.700000e+00	1.700000	3.250000e+
	25%	2.700000e+00	2.700000	3.700000e+
	50%	3.300000e+00	3.300000	4.400000e+
	75%	4.400000e+00	4.400000	4.600000e+
	max	6.500000e+00	6.500000	6.500000e+
Product_Weight_Grams	count	4.710000e+02	1259.000000	1.497000e+
	mean	1.725350e+02	170.008737	1.750053e+
	std	4.801394e+01	47.098220	6.289698e+
	min	9.000000e+01	90.000000	1.100000e+
	25%	1.500000e+02	150.000000	1.340000e+

	50%	1.700000e+02	170.000000	1.700000e+
	75%	1.750000e+02	175.000000	1.750000e+
	max	3.800000e+02	380.000000	3.800000e+
Product_Price_Per_100_Grams	**count**	4.710000e+02	1259.000000	1.497000e+
	mean	2.110573e+00	2.115338	2.642432e+
	std	6.691804e-01	0.671073	5.386785e-
	min	9.500000e-01	0.860000	8.600000e-
	25%	1.710000e+00	1.710000	2.510000e+
	50%	1.820000e+00	1.820000	2.760000e+
	75%	2.760000e+00	2.760000	3.070000e+
	max	3.450000e+00	3.450000	3.450000e+
Year	**count**	4.710000e+02	1259.000000	1.497000e+
	mean	2.018505e+03	2018.507546	2.018506e+
	std	5.005034e-01	0.500142	5.001268e-
	min	2.018000e+03	2018.000000	2.018000e+
	25%	2.018000e+03	2018.000000	2.018000e+
	50%	2.019000e+03	2019.000000	2.019000e+
	75%	2.019000e+03	2019.000000	2.019000e+
	max	2.019000e+03	2019.000000	2.019000e+
Month	**count**	4.710000e+02	1259.000000	1.497000e+
	mean	6.441614e+00	6.429706	6.440214e+
	std	3.383359e+00	3.504089	3.449686e+
	min	1.000000e+00	1.000000	1.000000e+
	25%	4.000000e+00	3.000000	3.000000e+
	50%	6.000000e+00	6.000000	6.000000e+
	75%	9.000000e+00	10.000000	9.000000e+
	max	1.200000e+01	12.000000	1.200000e+
Tax_Paid	**count**	4.710000e+02	1259.000000	1.497000e+
	mean	4.916348e-01	0.630548	7.886039e-
	std	2.209286e-01	0.205086	1.665676e-
	min	1.500000e-01	0.170000	3.000000e-
	25%	3.100000e-01	0.490000	6.700000e-
	50%	4.700000e-01	0.560000	7.600000e-
	75%	6.000000e-01	0.800000	8.400000e-
	max	1.380000e+00	1.530000	2.070000e+

	Life_Stage				Card_Subscription				
	count	unique	top	freq	count	unique	top	freq	co
Store_Number									
77	471	7	Young Singles/Couples	127	471	3	Mainstream	188	4
86	1259	7	Older Families	286	1259	3	Mainstream	535	12
88	1497	7	Older Singles/Couples	381	1497	3	Budget	534	14

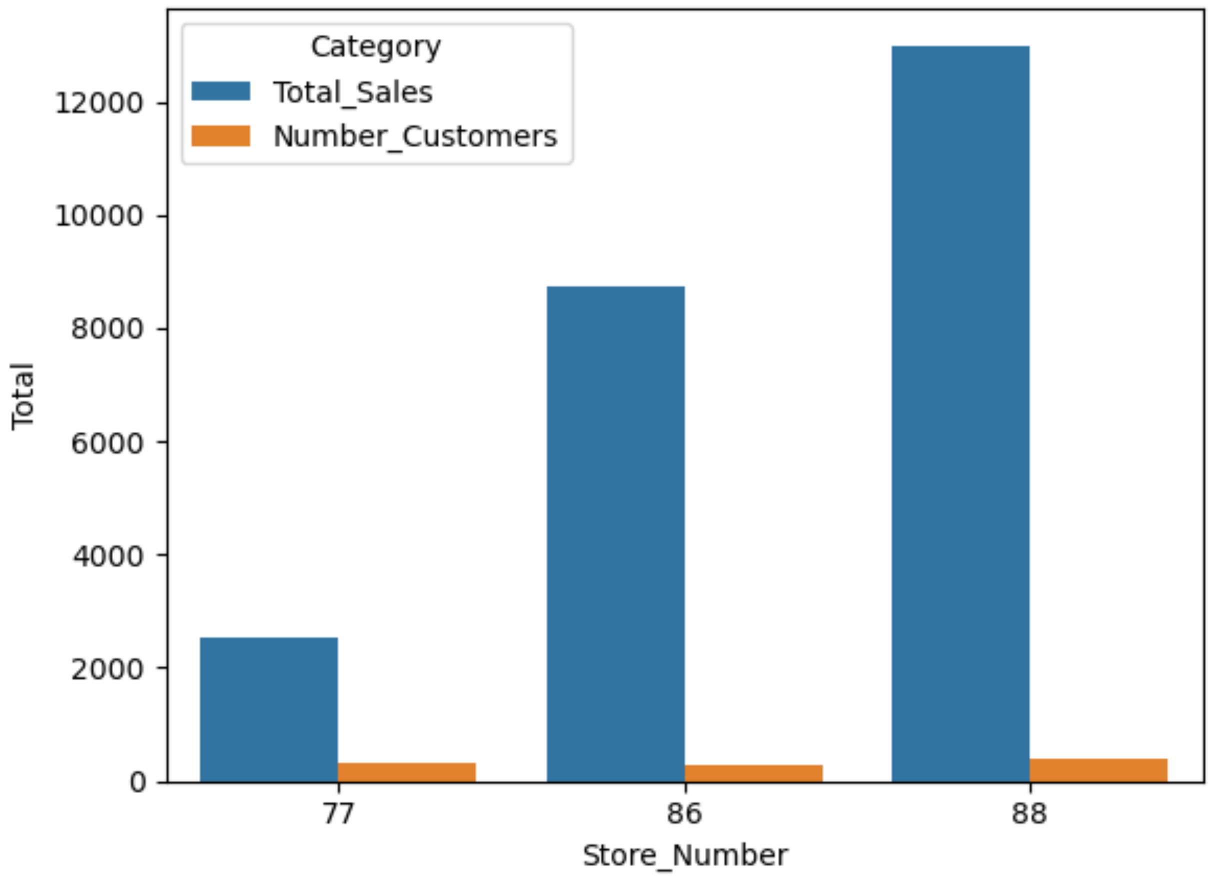
```
Set:  display.max_rows  to  10
```

In [766…

```
# Understanding the Stores of interest, based on sales experience.
    #total sales revenue
    #total number of customers
    #average number of transactions per customer

# Total Sales Revenue
print('Total Sales Revenue:')
display((
    data_set_df
        .query(f'Store_Number.isin({stores})')
        .groupby('Store_Number')
        ['Total_Sales']
        .sum()
))


# Total Number of Customers
print('Total Number of Customers:')

total_cust_df = (data_set_df
        .query(f'Store_Number.isin({stores})')
        .groupby(['Store_Number','Loyalty_Card_Number'])
        .count()
        .iloc[:,0:1]
       )

total_cust_df.iloc[:,0] = 1
total_cust_series = total_cust_df.unstack(1).sum(1)
display(total_cust_series)

# Average number of transactions per customer
print('Average number of transactions per customer:')
```

```
transaction_count_series = (
    data_set_df
        .query(f'Store_Number.isin({stores})')
        .groupby('Store_Number')
        ['Loyalty_Card_Number']
        .count()
)

display(round(transaction_count_series.divide(total_cust_series),2))

# Viusualisation.
print('--- Visualisation EDA ---')

total_sales_df = pd.DataFrame( data_set_df
                                    .query(f'Store_Number.isin({stores})'
                                    .groupby('Store_Number')
                                    ['Total_Sales']
                                    .sum())

total_cust_df = pd.DataFrame(total_cust_series).rename(columns={0:'Number



total_df = (pd
              .merge(left=total_sales_df,right=total_cust_df,on='Store_Num
              .reset_index()
              .melt(['Store_Number'])
              .rename(columns={'value':'Total','variable':'Category'})
            )

display(sns.barplot(data=total_df,x='Store_Number',y='Total',hue='Categor
```

```
Total Sales Revenue:
Store_Number
77     2546.30
86     8724.05
88    12995.45
Name: Total_Sales, dtype: float64
Total Number of Customers:
Store_Number
77    320.0
86    271.0
88    381.0
dtype: float64
Average number of transactions per customer:
Store_Number
77    1.47
86    4.65
88    3.93
dtype: float64
--- Visualisation EDA ---
<AxesSubplot:xlabel='Store_Number', ylabel='Total'>
```



3. Picking Control Stores

In [767…

```
# Ideally some changes are to occur in the trial stores.
# that will result in more sales, customers or some overall postive trend
# the experiment has concluded.

# In the scientific method a control group doesn't undergo any alteration
# Since, the trial stores may undergo transformations/changes.
# they need to be compared to a control group, for comparison.
# The control stores will hence, need to have similar attribute/twin like

# In this particular scenario, the control group chosen is a single store
# that has simlar metrics for sales, etc..

# The method for picking a control store.
# All non trial stores; are all possible control stores.
# To reduce it to 3 possible control stores,
# A process of elimination needs to be conducted based on some set of rul
# The Metrics: Average number of transactions per customer, Total Sales R

# Experiment/Change is assumed to be some event, process, postulation, th
# that will help in increasing sales, customer retention, customer transa
# and essentially has a positive outcome.
  # Some examples:
    # A discount, introduction of a new product
    # new chip brand introduced to consumers,
    # lighting conditions changes in store,
    # increase in staff,
    # renovation of store,
    # placement of products on shelves,
    # layout of store
    # furniture used in store
    # A new sub feature introduced in store: e.g. child care services whi
    # staff care more, new processes e.g. hand delivery to car/more conve
    # policy/management changes to store.
    # filling procedural changes based on conditions/events in daily life
    # unrelated e.g. time shifts of crew members. Even something unrelate
    # etc...

# Time period Selected as: Months;
    # Assumed.
    # Normal Period: first 6 months
    # Trial Period: the following 6 months.
    # Current data is fixed, new data needs to be collected for actual ch

        # Selecting a time period.
                # Since only two years, the months is a suitable time per
                #print(data_set_df['Month'].unique(), len(data_set_df['Mo
                #print(data_set_df['Year'].unique(), len(data_set_df['Yea

# Hypothesis
    # Null, Alternative. Formed to see differences.
    # Appropriate test conduted.

# Drivers: find reason for that change in metric to occur; assume a reaso
    # more customers, purchase per customer, more sales, more quantity bo
    # can be many, deduce some of them.
```

In [768…

```
# To find the control store that is similar to the trial store.
    # Pearson's correlation (positive r value, should have a y = x; simil
    # data_set_df.corr()

# Maybe: 155, 72, 237 # Fine, but not really.
# Other factors:
    # Hypothetical control ideal store for all trial stores.
    # Multiple control stores for a single trial store.
    # Using, another method for similarty deducing: not pearsn's correlat
```

In [769…

```
# A store is similar to another store:
    # Similar number of customers.
    # Same spending for avg transaction.
    # Store bring similar Total Sales.
    # Proportion of customer life stage is similar. (will utilise another
    # Note that filtering columns reduces the originality of comparison,
        # a fine tuning balance is required between the metrics of intere

def num_cust(s):
    return len(s.unique())

def avg_transact(s):
    return sum(s) / num_cust(s)



display(
    data_set_df
            .assign(Cust_Sub = data_set_df['Card_Subscription'])
            .groupby('Store_Number')
            .agg([num_cust,sum,avg_transact])
            [['Loyalty_Card_Number','Product_Number','Product_Quantity',
            .T
            .corr()
            .loc[:,stores]
)

# The correlation is too similar for each store. # Hence, this won't be u
# Another alternative is kept; which looks at more aggregation methods, v
```

```
/var/folders/2r/rg0cy7hn56970hbk66__swyr0000gn/T/ipykernel_6195/399406598
1.py:18: FutureWarning: ['Life_Stage', 'Card_Subscription', 'Age_Group',
'Relationship_Type', 'Date', 'Product_Name', 'Brand_Name', 'Cust_Sub'] di
d not aggregate successfully. If any error is raised this will raise in a
future version of pandas. Drop these columns/ops to avoid this warning.
  data_set_df
```

Store_Number	77	86	88
Store_Number			
1	0.990533	0.990646	0.990621
2	0.997008	0.997071	0.997057
3	0.998400	0.998446	0.998436
4	0.999134	0.999167	0.999161
5	0.999443	0.999471	0.999464
...	...	...	...
268	1.000000	0.999999	0.999999
269	0.999999	0.999999	0.999998
270	0.999999	0.999999	0.999998
271	0.999999	0.999999	0.999998
272	1.000000	0.999999	0.999999

271 rows × 3 columns

In [770…

```
# Assuming similarity with just all numeric columns.
    # This code is usable;  But, if a better measure is present, than tha
    # Still can be utilised.

store_determined = namedtuple('store_determined', 'trial_store control_st
sorter = lambda df,n:[store_determined(str_nm,df.sort_values(str_nm,ascen


display((
    data_set_df
        .groupby('Store_Number')
        .describe()
        .T
        .corr()
        .loc[:,stores]
        .pipe(sorter,1)
))
```

```
[store_determined(trial_store=77, control_store=155, pearson_correlation=
0.9236987331755908),
 store_determined(trial_store=86, control_store=71, pearson_correlation=0
.9605501073626765),
 store_determined(trial_store=88, control_store=237, pearson_correlation=
0.7168592296333446)]
```

In [771…

```
# Proportion of customer life stage is similar.
# the following dataframe can aid in deducing the similarity of proportio

# Used for rendering the dataframe as needed.
#toggle_option('all','display.max_rows')
toggle_option('28','display.max_rows')

cust_prop_creator = lambda df: df.assign(Cust_Prop = df['Count'] / df['Su
def diff_add(df):
    df1 = pd.DataFrame(data=None,columns=['Trial_Store_Number','Control_S
    for store_num in stores:
        for col in df.columns:
            df1 = pd.concat((df1,pd.DataFrame(np.array((store_num,col,sum
    return df1



life_stage_prop_diff_df = (
    pd.merge(left=(data_set_df
                .groupby(['Store_Number','Card_Subscription'])
                [['Product_Number']]
                .count()
                .groupby(['Store_Number'])
                .sum()
                .reset_index()
                ),
         right=(data_set_df
                .groupby(['Store_Number','Card_Subscription'])
                [['Product_Number']]
                .count()
                .reset_index()),
         on='Store_Number',
         how='inner',
        ).set_index(['Store_Number','Card_Subscription'])
         .rename(columns={'Product_Number_x':'Sum','Product_Number_y':'Co
         .pipe(cust_prop_creator)
         .unstack(1)
         ['Cust_Prop']
         .T
         .fillna(0)
         .pipe(diff_add)
         .set_index(['Trial_Store_Number','Control_Store_Number'])
         .sort_values('Life_Stage_Diff_Num')
         .query('`Trial_Store_Number` != `Control_Store_Number`')
         .head(10)
)
display(life_stage_prop_diff_df)
print(life_stage_prop_diff_df.describe())
```

```
Set:  display.max_rows  to  28
```

		Life_Stage_Diff_Num
Trial_Store_Number	**Control_Store_Number**	
77.0	**63.0**	0.005696
86.0	**251.0**	0.009842
	6.0	0.011171
	130.0	0.012108
88.0	**227.0**	0.012405
	233.0	0.012568
	164.0	0.014306
77.0	**167.0**	0.015491
	173.0	0.016031
	129.0	0.016204

```
       Life_Stage_Diff_Num
count            10.000000
mean              0.012582
std               0.003225
min               0.005696
25%               0.011405
50%               0.012487
75%               0.015195
max               0.016204
```

In [772…

```
# The life stage difference amongst the control and trial stores are very
    # The max difference for proportions of customer subscription type, b
    # being only 1.48 (approx) for a trial and control store.
    # Hence, whilst there is a minute difference taking that into account
    # can be ignored or each proposed control store can be explored with
    # with the most similar/mimium life stage proporion being for store n
            # being 0.0057 (approx); showcases that again either to ignor
```

In [773…

```
# The chosen stores for the control are: 77:155, 86;237, 88:71.
# Experimentation: Similar stores.
    # How else could it have been compared ? Transactions occured on simi
```

4. Comparing Control Stores to Trial Stores Performance

In [774…

```
# Time period chosing:
        # The data presented, is for a financial year worth of data, as s
        # The time period chosen is months. The first 6 months are kept a
        # The next 6 months is when the changes have occurred.
    # The reasons behind this time period is described within the followi
        # Investigate the statistical difference for first 6 months betwe
        # Some experiment/change/etc. occurs in trial store.
        # Observe the stat difference for the next 6 months between trial
        # The 6 months were chosen, as in other industries
            # E.g. healthcare, it can take upto and even more than 14 mon
        # Additionally, this even time period can ensure that both equal
        # Hence, this seemed appropriate to evenly split prior to experim

data_set_df = (
```

```
    data_set_df
            .assign(Date=pd.to_datetime(data_set_df.Date))
            )

print('Financial Year:',
        '\n',
        '\tDate Beginning: ',
        data_set_df.Date.dt.date.min(),
        '\n',
        '\tDate End:',
         data_set_df.Date.dt.date.max(),
         "\nTime Difference:",
        data_set_df.Date.dt.date.max() - data_set_df.Date.dt.date.min())



# What defines performance ?. Here is a list that resonates to the notion
  # The expansion or elevation of the following:
    # The quantitiy of sales.
    # The number of customer transactions.
    # The number of customer consuming chips.
    # The average order amount per transaction.
    # The product quantity purchased.
    # The consumption of chips weight.
    # The purchasing behaviour for chips.

  # Musings, that relate.
    # Life Stage similar, business strategy aligns in a particular way ?
        #E.g. More younger consumers buying more chips, habit building lo
    # Movements between stores of a customer,
        # a customer sticks to buying from one store to another store 'ad
    # Does brand proportion performance have an impact on store performan
    # Maybe profit margin is higher: revenue - tax paid. (technically thi
    # Maybe each time period cumulatively has better performance ?
    # The comparison factor, for buying chips, what proportion are before
        # How does this change over time periods ?

    # Segmentation:
    # What gets a person to buy chips ? # Too many, plausible reason henc
        # Group customers by customer type: # E.g. of a customer segmenta
            # See Alteration of a customer type for that store compared t
            # See key performance indicators for each customer segment.

    # Randomness:
        #how random is it ? Is there any reason or no reason for the stor
    # Dark data: unknown that could have been utilised.


# A statistical test is used to determine if there is a statistical signi

# For the datasets the following can also provide additional granularity
    # The set of assumptions. (Outside the scope of this task.)
    # Is there an underlying probability distribution that can be utilise
    # The Confidence Interval, chosen; The Sample Size given. (Can influe
    # The Decision making process for Type I (choosing alpha) and Type II
    # The type of distribution(t/z/etc...). Based on the query type of st

    # The framework and algorithm used for conducting a statistical test.
        # The purpose of the statistical test.
        # A simple layman idea is used here (neither a framework, nor a c
            # This is showcased for each test conducted.
```

```
            # This takes into consideration the above concepts, but not i

    # Related to this context,
        #A statistical test can be used to compare two datasets based on
            # Central tendency measure (e.g. Mean) and the variablity (e.
            # A set of assumptions based on distribution of data, probabi
            # Distributions:
                #The problem statement relating to task 1, doesn't state
                # Whilst, inferring from the text (As taught in english c
                    # As Dataset of multiple stores is provided, for all
                        # The population standard deviation can be hypoth
                            # but since the scope of this project is to b
                        #Hence, z distribution can be used.
                        #However, since a feasible probability is present
                            # the t distribution will be used for the cen

    # For Central Tendency:
        # Independent sample t test.
        # Matched sample t test.

    # For variability
        # Chi Square test.
    # This applies for each question above. Hence, a large number of test
        # Combining it with 3 different trial and store pairs, this gives
        # To adequately deal with this, functions are created.
    # This is are the test used, currently based on know how.


# Splitting the dataset.
    # Into 6 months periods of two.
        # The time interval can be into 2 periods, of any time length.

splitting_point =(
            pd
            .date_range(start=re.sub('-','/',str(data_set_df.Date.dt.date
                        end=re.sub('-','/',str(data_set_df.Date.dt.date.m
            .mean()
)

toggle_option('5', 'display.max_rows')
store_mapper = {77: 155, 86:71, 88:237}
trial_stores_pre_trial_df = data_set_df.query('Date < @splitting_point &
trial_stores_post_trial_df = data_set_df.query('Date > @splitting_point &
control_stores_pre_trial_df = data_set_df.query('Date < @splitting_point
control_stores_post_trial_df = data_set_df.query('Date > @splitting_point

display(trial_stores_pre_trial_df)
display(trial_stores_post_trial_df)
display(control_stores_pre_trial_df)
display(control_stores_post_trial_df)
```

```
Financial Year:
        Date Beginning:  2018-07-01
        Date End: 2019-06-30
Time Difference: 364 days, 0:00:00
Set:  display.max_rows  to  5
```

	Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationsl
59130	77000	Midage Singles/Couples	Budget	Midage	Singles
59136	77004	Retirees	Budget	Retirees	
...	...	...	...	...	
213984	2370961	Older Families	Budget	Older	
213985	2373711	Young Singles/Couples	Mainstream	Young	Singles

1579 rows × 19 columns

	Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationsl
59129	77000	Midage Singles/Couples	Budget	Midage	Singles
59131	77001	Young Families	Mainstream	Young	
...	...	...	...	...	
213971	2330291	Older Singles/Couples	Mainstream	Older	Singles
213975	2330501	Older Singles/Couples	Budget	Older	Singles

1642 rows × 19 columns

	Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationsl
54405	71000	Young Families	Budget	Young	
54406	71000	Young Families	Budget	Young	
...	...	...	...	...	
213956	880711	Older Families	Budget	Older	
213957	883791	Older Singles/Couples	Mainstream	Older	Singles

2049 rows × 19 columns

	Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationsl
54409	71000	Young Families	Budget	Young	
54410	71000	Young Families	Budget	Young	
...	...	...	...	...	
213953	862501	Young Families	Budget	Young	
213958	883791	Older Singles/Couples	Mainstream	Older	Singles

1984 rows × 19 columns

In [860…

```
# There are many approaches to utilise.
# however, a scaling whilst considered, wasn't performed amongst the cont

# Statistical Test # As previously mentioned not a concrete algorithm for
# Helpful steps in a statistical test.
# Step 1: Make Null and alternative hypothesis
# Step 2: Select distribution, statistical test, sample statistc formula
# Step 3: Select alpha value.
# Step 4: Define a selection rule based on hypothesis test.
# Step 5: Express data presented/write it in formal words.
# Step 6: Calculate the statistic of interest.
# Step 7: Interpret the notion with a normal distribution/usually include
# Step 8: Conclude to fail to reject or rejection of the null hypothesis.

# Is there a statistically significant change in the quantity of chips co
# Post trial stage compare that to control post stage.

# Based on logic. The contorl stores didn't undergo changes. So post cont
# The problem is they also assume that the customers can't move between s
# A customer can buy bread in store A (eg. Coles Footscray) and than also
# THe thing  is this assumption isn't taken care of by the wya they did t
```

In [776…

```
# Is it an 'independent' t - test ?

# Not independent if:
    # Same Customer buys chips from the contol store and the trial store.
        # A further explantion is the following:
            # If a clinical trial was conducted with the same patient bei
                # This is weakly mutually exclusive.
                # As this a flaw in the study design.

# If there is even a single customer that purchased between stores.
# This can't be independent.
    # For Stores 88 and 237 the independent T test cann't be done.
        # As the data collected depends on each other.
    # As it's okay to do an independent T test for the other 2 paris of s
        # What happens with 88 and 237 ?
            # The dependent data (same customer) is removed. Allowing for

toggle_option('10','display.max_rows')

for k,v in store_mapper.items():
    t1 = (
        data_set_df
            .query(f'Store_Number == {k}')

    )

    t2 = (
        data_set_df
            .query(f'Store_Number == {v}')
    )

    print(f'-- Trial store: {k} Control Store: {v} --')

    out = []
    for x in t1.Loyalty_Card_Number.unique():
        if x in t2.Loyalty_Card_Number.unique():
            out.append(x)
    print(out)
```

```
Set:  display.max_rows  to  10


-- Trial store: 77 Control Store: 155 --
[]
-- Trial store: 86 Control Store: 71 --
[]
-- Trial store: 88 Control Store: 237 --
[237324]
```

In [777…

```python
for data_frame in [control_stores_pre_trial_df, control_stores_post_trial
    (data_frame
         .query("Store_Number in list(@store_mapper.items())[-1] & Loyalt
         .pipe(lambda df: display(df) or df)
         .pipe(lambda df: print(df.index) or df)
        # .pipe(lambda df: df.drop(list(df.index)))
    )

control_stores_pre_trial_df = (control_stores_pre_trial_df
         .drop([191257])
    )


control_stores_post_trial_df = (control_stores_post_trial_df
         .drop([191259])
    )


trial_stores_pre_trial_df = (trial_stores_pre_trial_df
         .drop([191258])
    )


print('\n-- For Independent T Test to be plausible, the dependent data va

for data_frame in [control_stores_pre_trial_df, control_stores_post_trial
    (data_frame
         .query("Store_Number in list(@store_mapper.items())[-1] & Loyalt
         .pipe(lambda df: display(df) or df)
    )
```

	Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationsh
191257	237324	Midage Singles/Couples	Mainstream	Midage	Singles

```
Int64Index([191257], dtype='int64')
```

	Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationsh
191259	237324	Midage Singles/Couples	Mainstream	Midage	Singles

```
Int64Index([191259], dtype='int64')
```

	Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationsh
191258	237324	Midage Singles/Couples	Mainstream	Midage	Singles

```
Int64Index([191258], dtype='int64')
```

Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationship_Type	D

```
Int64Index([], dtype='int64')

-- For Independent T Test to be plausible, the dependent data values were
removed.--
```

Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationship_Type	D

Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationship_Type	D

Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationship_Type	D

Loyalty_Card_Number	Life_Stage	Card_Subscription	Age_Group	Relationship_Type	D

In [854…

```
# As the operations to be performed are similar for the trial store and a
# A function is made that can be utilised on other control and trail stor

###
# Question 1:  Does sales improve, or are greater in the post trial stage
###


# Step 7: Interpret the notion with a normal distribution/usually include
# Step 8: Conclude to fail to reject or rejection of the null hypothesis.

# Step 1: Make null and alternative hypothesis
    # Self explanatory, when conducting a classical hypothesis test .
    # H0: total_sales_mean_post_control - total_sales_mean_post_trial <=
    # H1: total_sales_mean_post_control - total_sales_mean_post_trial > 0

print('-- For each step, please refer to the, comments. -- ')

print()
print('Step 1:')
print('H0: total_sales_mean_post_control - total_sales_mean_post_trial <=
print('H1: total_sales_mean_post_control - total_sales_mean_post_trial >
print()
# Step 2: Select distribution, statistical test, sample statistic formula
    # Distribution: t (explained above)
    # Statistical Test: one sided.
    # Sample Statistical Formula: x_diff - hypth_popu_mean_diff/sqrt(vari
    #
print('Step 2:')
print('Distribution: T test', 'Statistical Test: One Sided', 'Sample Stat
print()
# Step 3: Select an alpha value .

    # Probablity density function from right hand side.
    # A decision of the alpha value needs to be made based on:
        # The alpha value determines the type I error, reducing it increa
            # Trading off between these two
            # rejecting the null and supporting the alternative when the
            # failing to reject the null and rejecting the alternative hy

        # The likelihood of a value falling 2 to 3 stdv is a good estimat
            # due to std error of the sampling mean difference distributi

      # Context of the problem.
            # Here, the typical 0.05 value is chosen.
    # 0.05 (approximate)
print('Step 3:')
print('Alpha Value: 0.05')
print()
```

```
# Step 4: Define a selection rule based on hypothesis test.
    # Using calculus.
    # or using a t-value table(df, alpha value),
        # the critical t value, and using the formula the critical sample
        # or a p value.
    # If p_value is less than or eqaul 0.05 to than reject null hypothesi
     # If p
crit_t_value_from_t_table = 1.66449
print('Step 4:')
print('Using a t-table, other alternatives could be taken.')
print(f'For: df = {len(control_stores_post_trial_df) + len(trial_stores_p
                  \nCritical t value = {crit_t_value_from_t_table}(greate
print('')
#Step 5: Express data in formal words.

    # For post_control sample
       # Sample mean(symbol x bar in output capital x ): mean(control_sto
        # sample Standard deviation (symbol s) = stdv(control_stores_post
        # Size of sample (n) = len(control_stores_post_trial_df)
    # For post_trial sample
        # Sample mean(symbol x bar in output capital x ): mean(trial_stor
        # sample Standard deviation (symbol s) = stdv(trial_stores_post_t
        # Size of sample (n) = len(trial_stores_post_trial_df)

print('Step 5:')
print('Control Stores Post Trial (Sample 1).')
print(f'\tSample Mean(X): {control_stores_post_trial_df.Total_Sales.mean(
print(f'\tSample Standard Deviation(s): {control_stores_post_trial_df.Tot
print(f'\tSample Size(n): {len(control_stores_post_trial_df)}')
print('Trial Stores Post Trial (Sample 2).')
print(f'\tSample Mean(X): {trial_stores_post_trial_df.Total_Sales.mean()}
print(f'\tSample Standard Deviation(s): {trial_stores_post_trial_df.Total
print(f'\tSample Size(n): {len(trial_stores_post_trial_df)}')

# Step 6: Calculate Statistic of interest.
    # Using the Formula for the t statistic:
         #(sample_mean_difference -  hypth_population_mean_diff)/sqrt(var
         #the t value is calculated
         #it helps to imagine a normal distribution
            #with a right sided shaded region being the rejection region(
            # and the left region being the failt to rejection region.
        # Imagine where that t value resides in the distribution, intuiti


variance1 = 2.3922341457622185 ** 2
variance2 = 2.3816451813891444 ** 2
t_value = ((7.512884518406377 - 7.509774665042578) + (0 - 0))/((variance1

print('Step 6:')
print('t value:',t_value)
print()

# Step 7: Interpret the notion with a normal distribution/usually include
    # Theory needs to be understood, for proper interpretation of the t v

print('Step 7:')
print('A t value close to 0 means that there is very little standard devi
print()

print('Step 8:')
print('As in earlier steps a decision rule was defined to help reject or
```

```python
print('This now comes into fruition.')
print('Since the t value is: ',round(t_value,5),'which is less than the c
print('Hence, a failure to reject the null hypothesis, as the total sales
print('\nThis, concludes the sample t test for total sales(improvements,


# Scipy could have been used, but it was best to show the process of doig

#sp.stats.ttest_ind(a=control_stores_post_trial_df.Total_Sales,
#                    b=trial_stores_post_trial_df.Total_Sales,
#                   nan_policy='raise',
#                  alternative='greater')
```

```
-- For each step, please refer to the, comments. --

Step 1:
H0: total_sales_mean_post_control - total_sales_mean_post_trial <= 0
H1: total_sales_mean_post_control - total_sales_mean_post_trial > 0

Step 2:
Distribution: T test Statistical Test: One Sided Sample Statistical Formu
la:
 (sample_mean_difference -  hypth_population_mean_diff)/sqrt(variance_sam
ple_1 + variance_sample_2)

Step 3:
Alpha Value: 0.05

Step 4:
Using a t-table, other alternatives could be taken.
For: df = 3623, Sufficiently large(Limit approaching Infinity: on t table
); One sided
Critical t value = 1.66449(greater than)

Step 5:
Control Stores Post Trial (Sample 1).
        Sample Mean(X): 7.512884518406377
        Sample Standard Deviation(s): 2.3922341457622185
        Sample Size(n): 1983
Trial Stores Post Trial (Sample 2).
        Sample Mean(X): 7.509774665042578
        Sample Standard Deviation(s): 2.3816451813891444
        Sample Size(n): 1642
Step 6:
t value: 0.0009212604013796538

Step 7:
A t value close to 0 means that there is very little standard deviation f
rom the mean sample difference distribution.

Step 8:
As in earlier steps a decision rule was defined to help reject or fail to
reject the null hypothesis.
This now comes into fruition.
Since the t value is:  0.00092 which is less than the critical t value of
a (right tailed test) of: 1.66
Hence, a failure to reject the null hypothesis, as the total sales sample
mean differnece is very similar.

This, concludes the sample t test for total sales(improvements, See comme
nts)
```

In [ ]:

```
# Note a similar procedure can be followed for different stores pairs; wi
# Slight deviations will occur in new tests e.g. chi squared for the test
# But the overall process will be the same.
```

In [ ]:

```
# Improvements.
    # Learning Statistical Tests.
    # A better use of steps, suitable knowdlege.
    # More implementation of different tests.
# Make conclusions.
```

In [859…

```
# Side note/challenge
 # make a function that interpetes parameters in any order of input.
    # n parameters.
        # n! ways of having parameters rearranged.
        # Some software magic allowing for same name too ?
            # Not the same type of parameter, e.g. *args could be used; i
        # bad programming practice, the vision of a programming language,
            # The user can specify the function parameter order as a prog
            # Hence, a different idea was used.
        # Changed from can this be done, to not done, but just try it any

def the_function(b, a, c):
        pass

# Easy: Chooser
# 1 make a rearranger function. -> parameter rerranger.

# Modules: math, itertools, inspect etc. can be used. But the aim was to
# Combining these two and doing more addition of logic can help form the
    # However, no modules were used in the making of the chooser.
    #math.factorial(3)
    #list(itertools.permutations('abc'))

def recur_fact(n):
    if n == 0 or n == 1:
        return 1
    return recur_fact(n-1) * n

def rearranger(iter_like):
    a_list = list(iter_like) # Memory usage increases. # Could be improve
    len_list = len(a_list)
    n = recur_fact(len_list)
    idx2,idx1 = len_list - 1, len_list - 2
    out = {}
    while n >= 1:
        out[n] = a_list[:]
        print(a_list,n)
        a_list[idx2],a_list[idx1] = a_list[idx1],a_list[idx2]
        idx2 -= 1
        idx1 -= 1
        n -= 1
        if idx1 == -1:
            idx2,idx1 = len_list - 1, len_list - 2
    return out

def chooser(func_name:str,*func_parameters) -> None :
    choices = rearranger(func_parameters)
    captured_out = input('\nWhich parameter order would you prefer ? ')
    try:
```

```python
        captured_out = int(captured_out)
        if captured_out not in choices:
            raise
    except:
        return print('The Input should be an integer, given from the outp
    print('\nHere is the function:')
    print('def '+ func_name+ str(tuple(choices[captured_out])) + ':' + '\


chooser('normalised_value','data_value', 'population_mean', 'population_s
#rearranger(['data_value','population_mean','population_stdv','out_messag
```

```
['data_value', 'population_mean', 'population_stdv'] 6
['data_value', 'population_stdv', 'population_mean'] 5
['population_stdv', 'data_value', 'population_mean'] 4
['population_stdv', 'population_mean', 'data_value'] 3
['population_mean', 'population_stdv', 'data_value'] 2
['population_mean', 'data_value', 'population_stdv'] 1

Which parameter order would you prefer ? 3

Here is the function:
def normalised_value('population_stdv', 'population_mean', 'data_value'):
        pass
```